9.1

IBM MQ

Nota

Antes de utilizar esta información y el producto al que se refiere, lea la información en “Avisos” en la página 2247.

Esta edición se aplica a la versión 9 release 1 de IBM® MQ y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

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

© Copyright International Business Machines Corporation 2007, 2024.

Contenido

Guía de consulta para el desarrollo de aplicaciones..... 7
Referencia de aplicaciones MQI..... 7
Ejemplos de código..... 8
Constantes..... 61
Tipos de datos utilizados en la MQI..... 235
Llamadas de funciones..... 632
objeto, atributos de..... 810
Códigos de retorno..... 889
Reglas para validar las opciones de MQI..... 890
Mensajes del mandato de publicación/suscripción en cola..... 893
Codificaciones de máquina..... 916
Opciones de informe y distintivos de mensaje..... 919
salida de conversión de datos..... 923
Propiedades especificadas como elementos MQRFH2..... 947
Conversión de páginas de códigos..... 956
Estándares de codificación en plataformas de 64 bits..... 1011
IBM i Application Programming Reference (ILE/RPG)..... 1015
Descripciones de tipos de datos en IBM i..... 1016
Llamadas de función en IBM i..... 1279
Atributos de objetos en IBM i..... 1401
Aplicaciones..... 1449
Códigos de retorno para IBM i (ILE RPG)..... 1462
Reglas para validar opciones MQI para IBM i (ILE RPG)..... 1464
Codificaciones de máquina en IBM i..... 1466
Opciones de informe y distintivos de mensaje en IBM i..... 1469
Conversión de datos en IBM i..... 1473
Proceso de conversión en IBM i..... 1473
Convenios de proceso en IBM i..... 1475
Conversión de mensajes de informe en IBM i..... 1479
MQDXP (parámetro de salida de conversión de datos) en IBM i..... 1480
MQXCNVC (Convertir caracteres) en IBM i..... 1485
MQCONVX (Salida de conversión de datos) en IBM i..... 1490
Referencia de salidas de usuarios, salidas de API y servicios instalables..... 1493
Estructura MQIEP..... 1494
Referencia de salida de conversión de datos..... 1497
MQ_PUBLISH_EXIT - Salida de publicación..... 1501
Llamadas de salida de canal y estructuras de datos..... 1509
Llamada de salida de carga de trabajo del clúster y estructuras de datos..... 1576
Referencia a la salida de la API..... 1602
Información de consulta sobre la interfaz de servicios instalables..... 1664
Información de consulta sobre la interfaz de servicios instalables en IBM i..... 1729
Las clases e interfaces de IBM MQ .NET..... 1769
Clase `MQAsyncStatus`.NET..... 1769
Clase `MQAuthenticationInformationRecord`.NET..... 1770
Clase `MQDestination`.NET..... 1771
Clase `MQEnvironment`.NET..... 1774
Clase `MQException`.NET..... 1776
Clase `MQGetMessageOptions`.NET..... 1777
Clase `MQManagedObject`.NET..... 1780
Clase `MQMessage`.NET..... 1783
Clase `MQProcess`.NET..... 1795
Clase `MQPropertyDescriptor`.NET..... 1797

Clase MQPutMessageOptions.NET 1799
Clase MQQueue.NET 1801
Clase MQQueueManager.NET 1809
Clase MQSubscription.NET 1822
Clase MQTopic.NET 1823
Interfaz de IMQObjectTrigger.NET 1829
Interfaz de MQC.NET 1830
Identificadores de juego de caracteres para aplicaciones .NET 1830
Clases C++ de IBM MQ 1833
Referencia cruzada de C++ y MQI 1834
Clase C++ de registro ImqAuthentication 1852
Clase C++ ImqBinary 1854
Clase C++ ImqCache 1856
Clase C++ ImqChannel 1859
ImqCICSBridgeHeader clase C++ 1864
Clase C++ ImqDeadLetterHeader 1870
ImqDistributionListar clase C++ 1873
Clase C++ ImqError 1874
Clase C++ ImqGetMessageOptions 1875
Clase C++ ImqHeader 1879
ImqIMSBridgeHeader clase C++ 1880
Clase C++ ImqItem 1883
Clase C++ ImqMessage 1885
Clase C++ de rastreador ImqMessage 1892
Clase C++ ImqNamelist 1895
Clase C++ ImqObject 1896
Clase C++ ImqProcess 1902
Clase C++ ImqPutMessageOptions 1904
Clase C++ ImqQueue 1906
Clase C++ del gestor de ImqQueue 1917
Clase C++ de cabecera ImqReference 1933
Clase C++ ImqString 1936
Clase C++ ImqTrigger 1941
Clase C++ de cabecera ImqWork 1944
Propiedades de objetos IBM MQ classes for JMS 1946
Dependencias entre propiedades de objetos IBM MQ classes for JMS 1950
APPLICATIONNAME 1952
ASYNCEXCEPTION 1953
BROKERCCDURSUBQ 1954
BROKERCCSUBQ 1954
BROKERCONQ 1954
BROKERDURSUBQ 1955
BROKERPUBQ 1955
BROKERPUBQMGR 1956
BROKERQMGR 1956
BROKERSUBQ 1957
BROKERVER 1957
CCDTURL 1958
CCSID 1958
CHANNEL 1959
CLEANUP 1959
CLEANUPINT 1960
ConnectionNameList 1960
CLIENTRECONNECTOPTIONS 1961
CLIENTRECONNECTTIMEOUT 1962
CLIENTID 1962
CLONESUPP 1963
COMPHDR 1963

COMPMSG.....1963
CONNOPT.....1964
CONNTAG.....1965
description.....1965
DIRECTAUTH.....1966
ENCODING.....1966
EXPIRY.....1967
FAILIFQUIESCE.....1968
HOSTNAME.....1969
LOCALADDRESS.....1969
ESTILO de nombre de mapa.....1970
MAXBUFFSIZE.....1970
MDREAD.....1971
MDWRITE.....1971
MDMSGCTX.....1972
MSGBATCHSZ.....1973
MSGBODY.....1973
MSGRETENTION.....1974
MSGSELECTION.....1974
MULTICAST.....1975
OPTIMISTICPUBLICATION.....1975
OUTCOMENOTIFICATION.....1976
PERSISTENCE.....1976
POLLINGINT.....1977
PORT.....1977
PRIORITY.....1978
PROCESSDURATION.....1978
PROVIDERVERSION.....1979
PROXYHOSTNAME.....1981
PROXYPORT.....1982
PUBACKINT.....1982
PUTASYNCALLOWED.....1983
QMANAGER.....1983
QUEUE.....1984
READAHEADALLOWED.....1984
READAHEADCLOSEPOLICY.....1985
RECEIVECCSID.....1985
RECEIVECONVERSION.....1986
RECEIVEISOLATION.....1986
RECEXIT.....1987
RECEXITINIT.....1987
REPLYTOSTYLE.....1988
RESCANINT.....1988
SECEXIT.....1989
SECEXITINIT.....1990
SENDCHECKCOUNT.....1990
SENDEXIT.....1990
SENDEXITINIT.....1991
SHARECONVALLOWED.....1991
SPARSESUBS.....1992
SSLCIPHERSUITE.....1993
SSLCRL.....1993
SSLFIPSREQUIRED.....1994
SSLPEERNAME.....1994
SSLRESETCOUNT.....1995
STATREFRESHINT.....1995
SUBSTORE.....1996
SYNCPOINTALLGETS.....1996

TARGCLIENT 1997
TARGCLIENTMATCHING 1997
TEMPMODEL 1998
TEMPQPREFIX 1998
TEMPTOPICPREFIX 1999
Tema 1999
TRANSPORT 1999
WILDCARDFORMAT 2000
La propiedad ENCODING 2001
Propiedades TLS de los objetos JMS 2001
Referencia de IBM Message Service Client for .NET 2002
.NET interfaces 2002
Propiedades de objetos XMS 2085
Referencia de aplicaciones en desarrollo de Managed File Transfer 2154
Ejemplos de uso de fteCreateTransfer para iniciar programas 2154
`fteAnt`: ejecutar tareas Ant en MFT 2156
Salidas de usuario de MFT para referencia de personalización 2181
Formatos de mensaje para los mensajes que puede colocar en la cola de mandatos del agente de MFT 2222
Referencia de la REST API de mensajería 2222
Recursos de REST API 2222

Avisos 2247
Información acerca de las interfaces de programación 2248
Marcas registradas 2249

Guía de consulta para el desarrollo de aplicaciones

Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las aplicaciones de IBM MQ .

- “Referencia de aplicaciones MQI” en la página 7
- IBM i “IBM i Application Programming Reference (ILE/RPG)” en la página 1015
- IBM i “Conversión de datos en IBM i” en la página 1473
- “Referencia de salidas de usuarios, salidas de API y servicios instalables” en la página 1493
- “Las clases e interfaces de IBM MQ .NET” en la página 1769
- “Clases C++ de IBM MQ” en la página 1833
- “Propiedades de objetos IBM MQ classes for JMS” en la página 1946
- V 9.1.0 “Referencia de la REST API de mensajería” en la página 2222

Tareas relacionadas

Desarrollo de aplicaciones

Referencia relacionada

Clases de IBM MQ para bibliotecas Java

Clases IBM MQ para JMS

Referencia de aplicaciones MQI

Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las aplicaciones MQI (Message Queue Interface).

- “Ejemplos de código” en la página 8
- “Constantes” en la página 61
- “Tipos de datos utilizados en la MQI” en la página 235
- “Llamadas de funciones” en la página 632
- “objeto, atributos de” en la página 810
- “Códigos de retorno” en la página 889
- “Reglas para validar las opciones de MQI” en la página 890
- “Codificaciones de máquina” en la página 916
- “Opciones de informe y distintivos de mensaje” en la página 919
- “salida de conversión de datos” en la página 923
- “Propiedades especificadas como elementos MQRFH2” en la página 947
- “Conversión de páginas de códigos” en la página 956

Conceptos relacionados

“Referencia de salidas de usuarios, salidas de API y servicios instalables” en la página 1493

Utilice la información de esta sección para ayudarle a desarrollar las salidas de usuario, las salidas de API y las aplicaciones de servicios instalables:

Tareas relacionadas

Desarrollo de aplicaciones

Referencia relacionada

“Las clases e interfaces de IBM MQ .NET” en la página 1769

© Copyright IBM Corp. 2007, 2024

Las clases e interfaces de IBM MQ .NET se listan alfabéticamente. Se describen las propiedades, métodos y constructores.

"Clases C++ de IBM MQ" en la página 1833
Las clases C++ de IBM MQ encapsulan la interfaz de cola de mensajes (MQI) de IBM MQ . Hay un único archivo de cabecera C++, **imqi.hpp**, que cubre todas estas clases.

Las bibliotecas de IBM MQ Classes for Java

IBM MQ Clases para JMS

Ejemplos de código

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

Ejemplos de lenguaje C

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

conectar con un gestor de colas

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

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

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

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

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

desconectarse de un gestor de colas

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

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

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

Creación de una cola dinámica

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

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

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

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

   }
```

```
   else {
      /*---------------------------------------*/
      /* Build an error message to report the  */
      /* failure of the opening of the model   */
      /* queue                                 */
      /*---------------------------------------*/
      MQMErrorHandling( "OPEN TEMPQ", CompCode,
                        Reason );
      ErrorFound = TRUE;
   }
   return ErrorFound;
}
⋮
```

Apertura de una cola existente

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

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

```
   if ((CompCode != MQCC_OK) || (Reason != MQRC_NONE))
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQOPEN, CompCode, Reason);
      PrintLine(pBuff);
```

```
      RetCode = CSQ4_ERROR;
      goto AbnormalExit1;       /* disconnect processing */
      }
   ⋮
   } /* end of main */
```

cerrar una cola

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

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

Colocación de un mensaje utilizando MQPUT

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS) z/OS y Programas de ejemplo para 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 */
        }
}
```

Colocación de un mensaje utilizando MQPUT1

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

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

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

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

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

```
void Process_Query(void)
   {
   /*                                                     */
   /* Build the reply message                             */
   /*                                                     */
   ⋮
   /*                                                     */
   /* Set the object descriptor, message descriptor and  */
   /* put message options to the values required to      */
```

```
   /* create the reply message.                          */
   /*                                                    */
   strncpy(ObjDesc.ObjectName, MsgDesc.ReplyToQ,
           MQ_Q_NAME_LENGTH);
   strncpy(ObjDesc.ObjectQMgrName, MsgDesc.ReplyToQMgr,
           MQ_Q_MGR_NAME_LENGTH);
   MsgDesc.MsgType = MQMT_REPLY;
   MsgDesc.Report  = MQRO_NONE;
   memset(MsgDesc.ReplyToQ, ' ', MQ_Q_NAME_LENGTH);
   memset(MsgDesc.ReplyToQMgr, ' ', MQ_Q_MGR_NAME_LENGTH);
   memcpy(MsgDesc.MsgId, MQMI_NONE, sizeof(MsgDesc.MsgId));
   PutMsgOpts.Options = MQPMO_SYNCPOINT +
                        MQPMO_PASS_IDENTITY_CONTEXT;
   PutMsgOpts.Context = Hobj_CheckQ;
   PutBuffLen = sizeof(PutBuffer);
   MQPUT1(Hconn,
          &ObjDesc,
          &MsgDesc,
          &PutMsgOpts,
          PutBuffLen,
          &PutBuffer,
          &CompCode,
          &Reason);

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

obtener un mensaje

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

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

```
                        MQGMO_BROWSE_FIRST +
                        MQGMO_ACCEPT_TRUNCATED_MSG;
   /*                                                   */
   /* Get the first message.                            */
   /*   Test for the output of the call is carried out  */
   /*   in the 'for' loop.                              */
   /*                                                   */
   MQGET(Hconn,
         Hobj,
         &MsgDesc,
         &GetMsgOpts,
         BufferLength,
         Buffer,
         &DataLength,
         &CompCode,
         &Reason);
```

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

Obtención de un mensaje utilizando la opción de espera

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

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

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

```
⋮
void main(void)
   {
   ⋮
   /*                                                   */
   /* Initialize options and open the queue for input   */
   /*                                                   */
   ⋮
```

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

Obtención de un mensaje utilizando la señalización

La señalización sólo está disponible con IBM MQ for z/OS .

Este ejemplo muestra cómo utilizar la llamada MQGET para establecer una señal para que se le notifique cuando llegue un mensaje adecuado a una cola. Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

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

```
if (signal_sw == 1)
   {
    endloop = 0;
```

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

Consulta de los atributos de un objeto

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

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CCC1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

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

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

```
      /*                                                  */
      /*   Issue the inquire call                         */
      /*     Test the output of the inquire call. If the */
      /*     call failed, display an error message       */
      /*     showing the completion code and reason code,*/
      /*     otherwise display the status of the         */
      /*     INHIBIT-GET and INHIBIT-PUT attributes      */
      /*                                                  */
      MQINQ(Hconn,
            *pHobj,
            SelectorCount,
            SelectorsTable,
            IntAttrCount,
            IntAttrsTable,
            CharAttrLength,
            CharAttrs,
            &CompCode,
            &Reason);
      if (CompCode != MQCC_OK)
         {
         sprintf(Message, MESSAGE_4_E,
                 ERROR_IN_MQINQ, CompCode, Reason);
         SetMsg(Message);
         }
      else
         {
           /* Process the changes */
         } /* end if CompCode */
```

Establecimiento de los atributos de una cola

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

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CCC1) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

const MQHCONN Hconn = MQHC_DEF_HCONN;

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

```
      SelectorsTable[1] = MQIA_INHIBIT_PUT;
      IntAttrsTable[0]  = MQQA_GET_INHIBITED;
      IntAttrsTable[1]  = MQQA_PUT_INHIBITED;
      ⋮
```

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

Recuperación de información de estado con MQSTAT

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

Este extracto se toma de la aplicación de ejemplo Llamando a MQSTAT (programa amqsapt0) suministrados con sistemas IBM MQ for Windows . Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

```
/*******************************************************************/
/*                                                                 */
/* Program name: AMQSAPT0                                          */
/*                                                                 */
/* Description: Sample C program that asynchronously puts messages  */
/*              to a message queue (example using MQPUT & MQSTAT).  */
/*                                                                 */
/* Licensed Materials - Property of IBM                            */
/*                                                                 */
/* 63H9336                                                         */
/* (c) Copyright IBM Corp. 2006, 2024. All Rights Reserved.        */
/*                                                                 */
/* US Government Users Restricted Rights - Use, duplication or     */
/* disclosure restricted by GSA ADP Schedule Contract with         */
/* IBM Corp.                                                       */
/*                                                                 */
/*******************************************************************/
/*                                                                 */
/* Function:                                                       */
/*                                                                 */
/*   AMQSAPT0 is a sample C program to put messages on a message   */
/*   queue with asynchronous response option, querying the success  */
/*   of the put operations with MQSTAT.                            */
/*                                                                 */
/*      -- messages are sent to the queue named by the parameter   */
/*                                                                 */
/*      -- gets lines from StdIn, and adds each to target          */
/*         queue, taking each line of text as the content          */
/*         of a datagram message; the sample stops when a null     */
/*         line (or EOF) is read.                                  */
/*         New-line characters are removed.                        */
```

```
/*         If a line is longer than 99 characters it is broken up   */
/*         into 99-character pieces. Each piece becomes the         */
/*         content of a datagram message.                           */
/*         If the length of a line is a multiple of 99 plus 1, for  */
/*         example, 199, the last piece will only contain a         */
/*         new-line character so will terminate the input.          */
/*                                                                  */
/*      -- writes a message for each MQI reason other than          */
/*         MQRC_NONE; stops if there is a MQI completion code       */
/*         of MQCC_FAILED                                           */
/*                                                                  */
/*      -- summarizes the overall success of the put operations     */
/*         through a call to MQSTAT to query MQSTAT_TYPE_ASYNC_ERROR*/
/*                                                                  */
/*    Program logic:                                                */
/*         MQOPEN target queue for OUTPUT                           */
/*         while end of input file not reached,                     */
/*         .  read next line of text                                */
/*         .  MQPUT datagram message with text line as data         */
/*         MQCLOSE target queue                                     */
/*         MQSTAT connection                                        */
/*                                                                  */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/*   AMQSAPT0 has the following parameters                          */
/*       required:                                                  */
/*                 (1) The name of the target queue                 */
/*       optional:                                                  */
/*                 (2) Queue manager name                           */
/*                 (3) The open options                             */
/*                 (4) The close options                            */
/*                 (5) The name of the target queue manager         */
/*                 (6) The name of the dynamic queue                */
/*                                                                  */
/********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
   /* includes for MQI */
#include <cmqc.h>

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

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

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

  /******************************************************************/
  /*                                                                */
  /*   Connect to queue manager                                     */
  /*                                                                */
  /******************************************************************/
  QMName[0] = 0;    /* default */
  if (argc > 2)
    strcpy(QMName, argv[2]);
  MQCONN(QMName,                  /* queue manager                 */
         &Hcon,                   /* connection handle             */
```

```
       &Compcode,               /* completion code                */
       &Reason);                /* reason code                    */
/* report reason and stop if it failed     */
if (CompCode == MQCC_FAILED)
{
  printf("MQCONN ended with reason code %d\n", CReason);
  exit( (int)CReason );
}

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

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

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

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

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

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

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

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

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

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

```
  pmo.Options |= MQPMO_ASYNC_RESPONSE;

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

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

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

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

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

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

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

  /*****************************************************************/
  /*                                                               */
  /*   Query how many asynchronous puts succeeded                  */
  /*                                                               */
```

```
  /*****************************************************************/
  MQSTAT(&Hcon,                   /* connection handle            */
         MQSTAT_TYPE_ASYNC_ERROR, /* status type                  */
         &Sts,                    /* MQSTS structure              */
         &CompCode,               /* completion code              */
         &Reason);                /* reason code                  */

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

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

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

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

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

Ejemplos de COBOL

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

conectar con un gestor de colas

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

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

```
* -------------------------------------------------------*
```

```
WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*    W02 - Data fields derived from the PARM field
 01  W02-MQM                     PIC X(48) VALUE SPACES.
*    W03 - MQM API fields
 01  W03-HCONN                   PIC S9(9) BINARY.
 01  W03-COMPCODE                PIC S9(9) BINARY.
 01  W03-REASON                  PIC S9(9) BINARY.
*
*    MQV contains constants (for filling in the control
*    blocks)
*    and return codes (for testing the result of a call)
*
 01  W05-MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
 ⋮
*    Separate into the relevant fields any data passed
*    in the PARM statement
*
     UNSTRING PARM-STRING DELIMITED BY ALL ','
                             INTO W02-MQM
                                  W02-OBJECT.
     ⋮
*    Connect to the specified queue manager.
*
     CALL 'MQCONN' USING W02-MQM
                         W03-HCONN
                         W03-COMPCODE
                         W03-REASON.
*
*    Test the output of the connect call.  If the call
*    fails, print an error message showing the
*    completion code and reason code.
*
     IF (W03-COMPCODE NOT = MQCC-OK) THEN
     ⋮
     END-IF.
     ⋮
```

desconectarse de un gestor de colas

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

Las variables utilizadas en esta extracción de código son las que se han establecido en “conectar con un gestor de colas” en la página 23. Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

Creación de una cola dinámica

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

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) proporcionada con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

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

```
*    Return to performing section.
*
     EXIT.
     EJECT
*
```

Apertura de una cola existente

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

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

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

```
*
*    Open the queue
*
     CALL 'MQOPEN' USING W02-HCONN
                         MQOD
                         W02-OPTIONS
                         W02-HOBJ
                         W02-COMPCODE
                         W02-REASON.
*
*    Test the output from the open
*
*    If the completion code is not OK, display a
*    separate error message for each of the following
*    errors:
*
```

```
*  Q-MGR-NOT-AVAILABLE - MQM is not available
*  CONNECTION-BROKEN   - MQM is no longer connected to CICS
*  UNKNOWN-OBJECT-NAME - The queue does not exist
*  NOT-AUTHORIZED      - The user is not authorized to open
*                        the queue
*
* For any other error, display an error message
* showing the completion and reason codes
*
  IF W02-COMPCODE NOT = MQCC-OK
     EVALUATE TRUE
*
       WHEN W02-REASON = MQRC-Q-MGR-NOT-AVAILABLE
            MOVE M01-MESSAGE-6 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-CONNECTION-BROKEN
            MOVE M01-MESSAGE-6 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-UNKNOWN-OBJECT-NAME
            MOVE M01-MESSAGE-2 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-NOT-AUTHORIZED
            MOVE M01-MESSAGE-3 TO M00-MESSAGE
*
       WHEN OTHER
            MOVE 'MQOPEN'      TO M01-MSG4-OPERATION
            MOVE W02-COMPCODE  TO M01-MSG4-COMPCODE
            MOVE W02-REASON    TO M01-MSG4-REASON
            MOVE M01-MESSAGE-4 TO M00-MESSAGE
        END-EVALUATE
     END-IF.
 E-EXIT.
*
*    Return to performing section
*
     EXIT.
     EJECT
```

cerrar una cola

Este ejemplo muestra cómo utilizar la llamada MQCLOSE.

Las variables utilizadas en esta extracción de código son las que se han establecido en “conectar con un gestor de colas” en la página 23. Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

Colocación de un mensaje utilizando MQPUT

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

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) proporcionada con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

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

Colocación de un mensaje utilizando MQPUT1

Este ejemplo muestra cómo utilizar la llamada MQPUT1 .

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB5) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

```
*
*    API control blocks
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-PUT-MESSAGE-OPTIONS.
     COPY CMQPMOV.
*
* CMQV contains constants (for filling in the
* control blocks) and return codes (for testing
* the result of a call).
*
 01  MQM-MQV.
 COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
⋮
*    Get the request message.
⋮
* -------------------------------------------------------*
 PROCESS-QUERY SECTION.
* -------------------------------------------------------*
⋮
*    Build the reply message.
⋮
*
* Set the object descriptor, message descriptor and
* put-message options to the values required to create
* the message.
* Set the length of the message.
*
  MOVE MQMD-REPLYTOQ    TO MQOD-OBJECTNAME.
  MOVE MQMD-REPLYTOQMGR TO MQOD-OBJECTQMGRNAME.
  MOVE MQMT-REPLY       TO MQMD-MSGTYPE.
  MOVE SPACES           TO MQMD-REPLYTOQ.
  MOVE SPACES           TO MQMD-REPLYTOQMGR.
  MOVE LOW-VALUES       TO MQMD-MSGID.
  COMPUTE MQPMO-OPTIONS = MQPMO-SYNCPOINT +
                          MQPMO-PASS-IDENTITY-CONTEXT.
  MOVE W03-HOBJ-CHECKQ  TO MQPMO-CONTEXT.
  MOVE LENGTH OF CSQ4BQRM-MSG TO W03-BUFFLEN.
*
     CALL 'MQPUT1' USING W03-HCONN
                         MQOD
                         MQMD
                         MQPMO
                         W03-BUFFLEN
                         W03-PUT-BUFFER
                         W03-COMPCODE
```

```
                        W03-REASON.
     IF W03-COMPCODE NOT = MQCC-OK
         MOVE 'MQPUT1'          TO M02-OPERATION
         MOVE MQOD-OBJECTNAME   TO M02-OBJECTNAME
         PERFORM RECORD-CALL-ERROR
         PERFORM FORWARD-MSG-TO-DLQ
     END-IF.
*
```

obtener un mensaje

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

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) proporcionada con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

```
*
*    Set get-message options
*
  COMPUTE MQGMO-OPTIONS = MQGMO-SYNCPOINT +
                          MQGMO-ACCEPT-TRUNCATED-MSG +
                          MQGMO-NO-WAIT.
*
* Set msgid and correlid in MQMD to nulls so that any
* message will qualify.
```

```
* Set length to available buffer length.
*
     MOVE MQMI-NONE TO MQMD-MSGID.
     MOVE MQCI-NONE TO MQMD-CORRELID.
     MOVE LENGTH OF W03-GET-BUFFER TO W03-BUFFLEN.
*
     CALL 'MQGET' USING W03-HCONN
                        W03-HOBJ-RESPONSE
                        MQMD
                        MQGMO
                        W03-BUFFLEN
                        W03-GET-BUFFER
                        W03-DATALEN
                        W03-COMPCODE
                        W03-REASON.
     EVALUATE TRUE
         WHEN W03-COMPCODE NOT = MQCC-FAILED
         ⋮
*            Process the message
             ⋮
         WHEN (W03-COMPCODE = MQCC-FAILED AND
               W03-REASON = MQRC-NO-MSG-AVAILABLE)
                 MOVE M01-MESSAGE-9 TO M00-MESSAGE
                 PERFORM CLEAR-RESPONSE-SCREEN
*
         WHEN OTHER
             MOVE 'MQGET  '     TO M01-MSG4-OPERATION
             MOVE W03-COMPCODE  TO M01-MSG4-COMPCODE
             MOVE W03-REASON    TO M01-MSG4-REASON
             MOVE M01-MESSAGE-4 TO M00-MESSAGE
             PERFORM CLEAR-RESPONSE-SCREEN
     END-EVALUATE.
```

Obtención de un mensaje utilizando la opción de espera

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

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB5) que se proporciona con IBM MQ for z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas procedimentales de ejemplo (plataformas excepto z/OS).

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

```
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
⋮
*    Open input queue.
⋮
```

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

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

Obtención de un mensaje utilizando la señalización

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

La señalización sólo está disponible con IBM MQ for z/OS .

```
⋮
* -------------------------------------------------------*
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W00 - General work fields
 ⋮
 01  W00-WAIT-INTERVAL    PIC S9(09) BINARY VALUE 30000.
*
*    W03 - MQM API fields
*
 01  W03-HCONN            PIC S9(9) BINARY VALUE ZERO.
```

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

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

```
          PERFORM RECORD-CALL-ERROR
          MOVE W06-CALL-ERROR  TO W06-CALL-STATUS
  END-EVALUATE.
*
 PROCESS-SIGNAL-ACCEPTED-EXIT.
*    Return to performing section
     EXIT.
     EJECT
*
```

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

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

```
*
     EXIT.
     EJECT
*
     ⋮
```

Consulta de los atributos de un objeto

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

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

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

```
*
*    Inquire about the attributes.
*
     CALL 'MQINQ' USING W02-HCONN,
                        W02-HOBJ,
                        W02-SELECTORCOUNT,
                        W02-SELECTORS-TABLE,
                        W02-INTATTRCOUNT,
                        W02-INTATTRS-TABLE,
                        W02-CHARATTRLENGTH,
                        W02-CHARATTRS,
                        W02-COMPCODE,
                        W02-REASON.
*
* Test the output from the inquiry:
```

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

Establecimiento de los atributos de una cola

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

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

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

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

```
     MOVE MQQA-PUT-INHIBITED TO W02-INTATTRS(2).
*
*    Set the attributes.
*
     CALL 'MQSET' USING W02-HCONN,
                        W02-HOBJ,
                        W02-SELECTORCOUNT,
                        W02-SELECTORS-TABLE,
                        W02-INTATTRCOUNT,
                        W02-INTATTRS-TABLE,
                        W02-CHARATTRLENGTH,
                        W02-CHARATTRS,
                        W02-COMPCODE,
                        W02-REASON.
*
* Test the output from the call:
*
*  - If the completion code is not OK, display an error
*    message showing the completion and reason codes
*
*  - Otherwise, move 'INHIBITED' into the relevant
*    screen map fields
*
     IF W02-COMPCODE NOT = MQCC-OK
        MOVE 'MQSET'       TO M01-MSG4-OPERATION
        MOVE W02-COMPCODE  TO M01-MSG4-COMPCODE
        MOVE W02-REASON    TO M01-MSG4-REASON
        MOVE M01-MESSAGE-4 TO M00-MESSAGE
     ELSE
*
*       Process the changes.
⋮
     END-IF.
```

Ejemplos de lenguaje ensamblador de System/390

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

conectar con un gestor de colas

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

Este extracto se toma del programa de ejemplo de examen (CSQ4BAA1) proporcionado con IBM MQ for z/OS.

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

```
         LA    R4,MQMNAME      Address for target
         BCTR  R1,R0           Reduce for execute
         EX    R1,MOVEPARM     Move the data
*
*********************************************************
* EXECUTES                                              *
*********************************************************
MOVEPARM MVC   0(*-*,R4),0(R3)
*
         EJECT
```

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

desconectarse de un gestor de colas

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

```
BADCALL  DS    0H
⋮
*                  CONSTANTS
*
         CMQA
*
*        WORKING STORAGE (RE-ENTRANT)
*
WEG3     DSECT
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
```

```
*
HCONN    DS   F
COMPCODE DS   F
REASON   DS   F
*
*
LEG3     EQU  *-WKEG3
         END
```

Creación de una cola dinámica

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

```
         ⋮
         END
```

Apertura de una cola existente

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

Muestra cómo especificar dos opciones. Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

cerrar una cola

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

```
⋮
*
* ISSUE MQI CLOSE REQUEST USING REENTRANT FROM OF
```

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

Colocación de un mensaje utilizando MQPUT

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

     LA   R5,BUFFER_LEN   RETRIEVE THE BUFFER LENGTH
     ST   R5,BUFFLEN      AND SAVE IT FOR MQM USE
*
     MVC  BUFFER,TEST_MSG    SET THE MESSAGE TO BE PUT
```

```
*
*    ISSUE MQI PUT REQUEST USING REENTRANT FORM
*    OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL MQPUT,                          X
               (HCONN,                        X
               HOBJ,                          X
               WMD,                           X
               WPMO,                          X
               BUFFLEN,                       X
               BUFFER,                        X
               COMPCODE,                      X
               REASON),VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK
         C   R5,COMPCODE
         BNE BADCALL
*        ⋮
BADCALL DS  0H
⋮
```

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

Colocación de un mensaje utilizando MQPUT1

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

```
⋮
*
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
PUT      DS  0H
*
         MVC  WOD_AREA,MQOD_AREA     INITIALIZE WORKING VERSION OF
*                                    MQOD WITH DEFAULTS
         MVC  WOD_OBJECTNAME,Q_NAME  SPECIFY Q NAME FOR PUT1
*
```

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

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

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

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

obtener un mensaje

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

```
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
DATALEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
*
BUFFER   DS CL80
BUFFER_LEN EQU *-BUFFER
*
```

```
WMD      CMQMDA DSECT=NO,LIST=NO
WGMO     CMQGMOA DSECT=NO,LIST=NO
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
⋮
END
```

Obtención de un mensaje utilizando la opción de espera

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

Este código acepta mensajes truncados. Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

```
         C   R5,REASON                        MESSAGE?
         BE  GETOK                   YES, SO GO AND PROCESS.
         B   BADCALL                 NO, SOME OTHER WARNING
*
NOMSG    DS  0H
         ⋮
GETOK    DS  0H
         ⋮
```

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

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

Obtención de un mensaje utilizando la señalización

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

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

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

Consulta y establecimiento de los atributos de una cola

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

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

Ejemplos de PL/I

El uso de PL/I solo está soportado por z/OS . Esta colección de temas muestra técnicas que utilizan ejemplos de PL/I.

conectar con un gestor de colas

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

desconectarse de un gestor de colas

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

Creación de una cola dinámica

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

Apertura de una cola existente

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

cerrar una cola

Este ejemplo muestra cómo utilizar la llamada MQCLOSE.

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

Colocación de un mensaje utilizando MQPUT

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

```
            COMPCODE,
            REASON);
```

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

Colocación de un mensaje utilizando MQPUT1

Este ejemplo muestra cómo utilizar la llamada MQPUT1 .

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

CALL MQPUT1 (HCONN,
```

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

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

obtener un mensaje

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

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

```
                 DATALEN,
                 COMPCODE,
                 REASON);

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

Obtención de un mensaje utilizando la opción de espera

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

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

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

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

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

Obtención de un mensaje utilizando la señalización

Extracción de código que muestra cómo utilizar la llamada MQGET con señalización.

La señalización sólo está disponible con IBM MQ for z/OS .

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

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

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

```
              ⋮
           END;

          END;

        END;
     ⋮
 END DO_WORK;

 GET_MSG: PROC;
```

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

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

 END GET_MSG;

 NO_MSG: PROC;
 ⋮
 END NO_MSG;
```

Consulta de los atributos de un objeto

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

     SELECTORS(01) = MQIA_CURRENT_Q_DEPTH;

     SELECTORCOUNT  = 1;
     INTATTRCOUNT   = 1;

     CHARATTRLENGTH = 0;
/*********************************************************/
/*                                                       */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.             */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.              */
```

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

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

Establecimiento de los atributos de una cola

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

Este extracto no se toma de las aplicaciones de ejemplo suministradas con IBM MQ.

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

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

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

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

     SELECTORCOUNT  = 2;
     INTATTRCOUNT   = 2;

     CHARATTRLENGTH = 0;
```

```
/************************************************************/
/*                                                          */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                 */
/*                                                          */
/************************************************************/
     CALL MQSET (HCONN,
                 HOBJ,
```

```
                  SELECTORCOUNT,
                  SELECTORS,
                  INTATTRCOUNT,
                  INTATTRS,
                  CHARATTRLENGTH,
                  CHARATTRS,
                  COMPCODE,
                  REASON);

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

Constantes

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

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

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

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

Archivos de cabecera C

Los archivos de cabecera se proporcionan para ayudarle a escribir programas de aplicación C que utilizan la MQI.

Los archivos de cabecera C se resumen en la tabla siguiente:

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

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

Archivos de copia COBOL

Se proporcionan varios archivos COPY para ayudarle a escribir programas de aplicación COBOL que utilizan MQI.

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

Nombre de archivo	Descripción	IBM i	UNIX	Windows	z/OS
Códigos de retorno y constantes					
CMQx	Definiciones MQI	V	V	V	V
CMQCFx	PCF, definiciones	V	V	V	V
CMQPSx	Definiciones de publicación/ suscripción	V	V	V	V
CMQXx	Definiciones de canal y salida	V	V	V	V
Estructuras					
CMQAIRx	MQAIR-Registro de información de autenticación		V L	V L	
CMQBOx	MQBO-Opciones de inicio	V L	V L	V L	
CMQCDx	MQCD-Definición de canal	V L	V L	V L	V L
CMQCFBFx	MQCFBF-Parámetro de filtro de serie de bytes PCF	V L	V L	V L	V L
CMQCFBSx	MQCFBS-Parámetro de serie de bytes PCF	V L	V L	V L	V L
CMQCFGRx	MQCFGR-Parámetro de grupo PCF	V L	V L	V L	V L
CMQCFHx	MQCFH-cabecera PCF	V L	V L	V L	V L
CMQCFIFx	MQCFIF-Parámetro de filtro de enteros PCF	V L	V L	V L	V L
CMQCFILx	MQCFIL-Parámetro de lista de enteros PCF	V L	V L	V L	V L
CMQCFINx	MQCFIN-Parámetro entero PCF	V L	V L	V L	V L
CMQCFSFx	MQCFSF-Parámetro de filtro de serie PCF	V L	V L	V L	V L
CMQCFSLx	MQCFSL-Parámetro de lista de series PCF	V L	V L	V L	V L
CMQCFSTx	MQCFST-Parámetro de serie PCF	V L	V L	V L	V L
CMQCFXLx	MQCFIL64 -Parámetro de lista de enteros de PCF de 64 bits	V L	V L	V L	V L
CMQCFXNx	MQCFIN64 -Parámetro entero de PCF de 64 bits	V L	V L	V L	V L
CMQCHRVx	MQCHARV-Serie de longitud variable	V L	V L	V L	V L
CMQCIHx	MQCIH-Cabecera CICS bridge	V L	V L	V L	V L
CMQCNOx	MQCNO - Opciones de conexión	V L	V L	V L	V L

Tabla 2. Archivos de copia COBOL: códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQCSPx	MQCSP-Parámetros de seguridad	V L	V L	V L	V L
CMQCXPx	MQCXP-Parámetros de salida de canal	V L			V L
CMQDHx	MQDH - Cabecera de distribución	V L	V L	V L	V L
CMQDLHx	MQDLH-Cabecera de mensaje no entregado	V L	V L	V L	V L
CMQDXPx	MQDXP-Parámetros de salida de conversión de datos	V L		V L	
CMQEPHx	MQEPH - Cabecera PCF incrustada	V L	V L	V L	V L
CMQGMOx	MQGMO-Obtener opciones de mensaje	V L	V L	V L	V L
CMQIIHx	MQIIH – Cabecera información de IMS	V L	V L	V L	V L
CMQMDx	MQMD - Descriptor de mensaje	V L	V L	V L	V L
CMQMD1x	MQMD1 -Descriptor de mensaje versión 1	V L	V L	V L	V L
CMQMD2x	MQMD2 -Descriptor de mensaje versión 2	V L	V L	V L	V L
CMQMDEx	MQMDE-Descriptor de mensaje ampliado	V L	V L	V L	V L
CMQODx	MQOD-Descriptor de objeto	V L	V L	V L	V L
CMQORx	MQOR-Registro de objeto	V L	V L	V L	V L
CMQPMOx	MQPMO-Opciones de transferencia de mensajes	V L	V L	V L	V L
CMQRFHx	MQRFH - Cabecera de reglas y formato	V L	V L	V L	V L
CMQRFH2x	MQRFH2 – Reglas y formato de la cabecera 2	V L	V L	V L	V L
CMQRMHx	MQRMH - Cabecera de mensaje de referencia	V L	V L	V L	V L
CMQRRx	MQRR-Registro de respuesta	V L	V L	V L	
CMQSCOx	MQSCO-Opciones de configuración de TLS		V L	V L	
CMQTMx	MQTM-Mensaje de desencadenante	V L		V L	V L
CMQTMCx	MQTMC-Carácter de mensaje de desencadenante	V L	V L		
CMQTMC2x	MQTMC2 -Mensaje de desencadenante de 2 caracteres	V L	V L	V L	V L

Tabla 2. Archivos de copia COBOL: códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQWIHx	MQWIH - Cabecera de información de trabajo	V L	V L	V L	V L
CMQXQHx	MQXQH - Cabecera de cola de transmisión	V L	V L	V L	V L
Clave: • Archivos con valores iniciales proporcionados, x = V • Archivos sin valores iniciales proporcionados, x = L					

z/OS *Archivos de inclusión PL/I*

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

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

IBM i *Archivos de copia RPG*

Los archivos RPG COPY se proporcionan para el lenguaje de programación RPG. Estos archivos sólo están disponibles en IBM i.

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
Códigos de retorno y constantes					
CMQx	Definiciones MQI	G R			
CMQCFx	PCF, definiciones	G			
CMQPSx	Definiciones de publicación/ suscripción	G			
CMQXx	Definiciones de canal y salida	G R			
Estructuras					
CMQBOx	MQBO-Opciones de inicio	G H			

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQCDx	MQCD-Definición de canal	G H R			
CMQCFBFx	MQCFBF-Parámetro de filtro de serie de bytes PCF	G H			
CMQCFBSx	MQCFBS-Parámetro de serie de bytes PCF	G H			
CMQCFGRx	MQCFGR-Parámetro de grupo PCF	G H			
CMQCFHx	MQCFH-cabecera PCF	G H			
CMQCFIFx	MQCFIF-Parámetro de filtro de enteros PCF	G H			
CMQCFILx	MQCFIL-Parámetro de lista de enteros PCF	G H			
CMQCFINx	MQCFIN-Parámetro entero PCF	G H			
CMQCFSFx	MQCFSF-Parámetro de filtro de serie PCF	G H			
CMQCFSLx	MQCFSL-Parámetro de lista de series PCF	G H			
CMQCFSTx	MQCFST-Parámetro de serie PCF	G H			
CMQCFXLx	MQCFIL64 -Parámetro de lista de enteros de PCF de 64 bits	G H			
CMQCFXNx	MQCFIN64 -Parámetro entero de PCF de 64 bits	G H			
CMQCHARVx	MQCHARV-Serie de longitud variable	G H			
CMQCIHx	MQCIH-Cabecera CICS bridge	G H			
CMQCNOx	MQCNO - Opciones de conexión	G H			
CMQCSPx	MQCSP-Parámetros de seguridad	G H			
CMQCXPx	MQCXP-Parámetros de salida de canal	G H R			
CMQDHx	MQDH - Cabecera de distribución	G H R			
CMQDLHx	MQDLH-Cabecera de mensaje no entregado	G H R			
CMQDXPx	MQDXP-Parámetros de salida de conversión de datos	G H R			
CMQEPHx	MQEPH - Cabecera PCF incrustada	G H			
CMQGMOx	MQGMO-Obtener opciones de mensaje	G H R			
CMQIIHx	MQIIH – Cabecera información de IMS	G H R			
CMQMDx	MQMD - Descriptor de mensaje	G H R			

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQMD1x	MQMD1 -Descriptor de mensaje versión 1	G H R			
CMQMD2x	MQMD2 -Descriptor de mensaje versión 2	G H			
CMQMDEx	MQMDE-Descriptor de mensaje ampliado	G H R			
CMQODx	MQOD-Descriptor de objeto	G H R			
CMQORx	MQOR-Registro de objeto	G H R			
CMQPMOx	MQPMO-Opciones de transferencia de mensajes	G H R			
CMQPXPx	MQPXP-Parámetros de salida de direccionamiento de publicación/ suscripción	G H			
CMQRFHx	MQRFH - Cabecera de reglas y formato	G H			
CMQRFH2x	MQRFH2 – Reglas y formato de la cabecera 2	G H			
CMQRMHx	MQRMH - Cabecera de mensaje de referencia	G H R			
CMQRRx	MQRR-Registro de respuesta	G H R			
CMQTMx	MQTM-Mensaje de desencadenante	G H R			
CMQTMCx	MQTMC-Carácter de mensaje de desencadenante	G H R			
CMQTMC2x	MQTMC2 -Mensaje de desencadenante de 2 caracteres	G H R			
CMQWIHx	MQWIH - Cabecera de información de trabajo	G H			
CMQXQHx	MQXQH - Cabecera de cola de transmisión	G H R			

Clave:

- Archivo para enlace estático, inicializado, proporcionado x = G
- Archivo para enlace estático, no inicializado, proporcionado x = H
- Archivo para enlace dinámico, inicializado, proporcionado, x = R

Windows *Archivos de módulo Visual Basic*

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

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

z/OS *Archivos COPY de z/OS Assembler*

Se proporcionan varios archivos COPY para ayudarle a escribir programas de aplicación de z/OS Assembler que utilizan MQI.

Tabla 6. Archivos de copia de z/OS Assembler: tipos de datos, códigos de retorno, constantes y estructuras					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
Tipos de datos, códigos de retorno y constantes					
CMQA	Definiciones MQI				A
CMQCFA	PCF, definiciones				A
CMQPSA	Definiciones de publicación/ suscripción				A
CMQVERA	Control de versión de estructura				A
CMQXA	Definiciones de canal y salida				A
Estructuras					
CMQCDA	MQCD-Definición de canal				
CMQCFBFA	MQCFBF-Parámetro de filtro de serie de bytes PCF				
CMQCFBSA	MQCFBS-Parámetro de serie de bytes PCF				A
CMQCFGRA	MQCFGR-Parámetro de grupo PCF				A
CMQCFHA	MQCFH-cabecera PCF				A
CMQCFIFA	MQCFIF-Parámetro de filtro de enteros PCF				A
CMQCFILA	MQCFIL-Parámetro de lista de enteros PCF				A
CMQCFINA	MQCFIN-Parámetro entero PCF				A
CMQCFSFA	MQCFSF-Parámetro de filtro de serie PCF				A

Tabla 6. Archivos de copia de z/OS Assembler: tipos de datos, códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQCFSLA	MQCFSL-Parámetro de lista de series PCF				A
CMQCFSTA	MQCFST-Parámetro de serie PCF				A
CMQCFXLA	MQCFIL64 -Parámetro de lista de enteros de PCF de 64 bits				A
CMQCFXNA	MQCFIN64 -Parámetro entero de PCF de 64 bits				A
CMQCHARVA	MQCHARV-Serie de longitud variable				A
CMQCIHA	MQCIH-Cabecera CICS bridge				A
CMQCNOA	MQCNO - Opciones de conexión				A
CMQCSPA	MQCSP-Parámetros de seguridad				A
CMQCXPA	MQCXP-Parámetros de salida de canal				A
CMQDHA	MQDH - Cabecera de distribución				A
CMQDLHA	MQDLH-Cabecera de mensaje no entregado				A
CMQDXPA	MQDXP-Parámetros de salida de conversión de datos				A
CMQEPHA	MQEPH - Cabecera PCF incrustada				A
CMQGMOA	MQGMO-Obtener opciones de mensaje				A
CMQIIHA	MQIIH – Cabecera información de IMS				A
CMQMDA	MQMD - Descriptor de mensaje				A
CMQMD1A	MQMD1 -Descriptor de mensaje versión 1				A
CMQMD2A	MQMD2 -Descriptor de mensaje versión 2				A
CMQMDEA	MQMDE-Descriptor de mensaje ampliado				A
CMQODA	MQOD-Descriptor de objeto				A
CMQORA	MQOR-Registro de objeto				A
CMQPMOA	MQPMO-Opciones de transferencia de mensajes				A
CMQRFHA	MQRFH - Cabecera de reglas y formato				A

Tabla 6. Archivos de copia de z/OS Assembler: tipos de datos, códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**UNIX**	**Windows**	**z/OS**
CMQRFH2A	MQRFH2 – Reglas y formato de la cabecera 2				A
CMQRMHA	MQRMH - Cabecera de mensaje de referencia				A
CMQTMA	MQTM-Mensaje de desencadenante				A
CMQTMC2A	MQTMC2 -Mensaje de desencadenante de 2 caracteres				A
CMQWCRA	MQWCR-Registro de clúster de carga de trabajo de clúster				A
CMQWDRA	MQWDR-Registro de destino de carga de trabajo de clúster				A
CMQWDR1A	MQWDR1 -Registro de destino de carga de trabajo de clúster versión 1				A
CMQWDR2A	MQWDR2 -Registro de destino de carga de trabajo de clúster versión 2				A
CMQWIHA	MQWIH - Cabecera de información de trabajo				A
CMQWQRA	MQWQR-Registro de cola de carga de trabajo de clúster				A
CMQWQR1A	MQWQR1 -Registro de cola de carga de trabajo de clúster versión 1				A
CMQWQR2A	MQWQR2 -Registro de cola de carga de trabajo de clúster versión 2				A
CMQWXPA	MQWXP-Parámetros de salida de carga de trabajo de clúster				A
CMQWXP1A	MQWXP1 -Parámetros de salida de carga de trabajo de clúster versión 1				A
CMQWXP2A	MQWXP2 -Parámetros de salida de carga de trabajo de clúster versión 2				A
CMQWXP3A	MQWXP3 -Parámetros de salida de carga de trabajo de clúster versión 3				A
CMQXPA	MQXP-Parámetros de salida cruzada de API de CICS				A
CMQXQHA	MQXQH - Cabecera de cola de transmisión				A
CMQXWDA	MQXWD-Descriptor de espera de salida				A
Clave: A= Archivo proporcionado					

MQ_ * (longitudes de serie)

Tabla 7. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
MQ_APPL_FUNCTION_NAME_LENGTH	10	X'0000000A'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'
MQ_APPL_NAME_LENGTH	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'
MQ_APPL_TAG_LENGTH	28	X'0000001C'
MQ_LONGITUD_SUFIJO	2	X'00000002'
MQ_ATTENTION_ID_LENGTH	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_NAME_LENGTH	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH	8	X'00000008'
MQ_AUTO_REORG_CATALOG_LENGTH	44	X'0000002C'
MQ_AUTO_REORG_TIME_LENGTH	4	X'00000004'
MQ_BATCH_INTERFACE_ID_LENGTH	8	X'00000008'
MQ_BRIDGE_NAME_LENGTH	24	X'00000018'
MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
MQ_CHANNEL_NAME_LENGTH	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
MQ_LONGITUD_ID_CLIENTE	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH	48	X'00000030'
MQ_CONN_NAME_LENGTH	264	X'00000108'
LONGITUD_CÓDIGO_CONEXIÓN	128	X'00000080'
MQ_CONNECTION_ID_LENGTH	24	X'00000018'
MQ_CORREL_ID_LENGTH	24	X'00000018'
MQ_CREATION_DATE_LENGTH	12	X'0000000C'
MQ_CREATION_TIME_LENGTH	8	X'00000008'
MQ_DATE_LENGTH	12	X'0000000C'
MQ_NOMBRE_DISTINGUIDO_LONGITUD	1024	X'00000400'
MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'

Tabla 7. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_EXIT_DATA_LENGTH	32	X'00000020'
MQ_EXIT_INFO_NAME_LENGTH	48	X'00000030'
MQ_EXIT_NAME_LENGTH	(value differs by platform or version)	
MQ_EXIT_PD_AREA_LENGTH	48	X'00000030'
MQ_EXIT_USER_AREA_LENGTH	16	X'00000010'
MQ_FACILITY_LENGTH	8	X'00000008'
MQ_FACILITY_LONGITUD_CLAVE	4	X'00000004'
MQ_FORMAT_LENGTH	8	X'00000008'
LONGITUD_FUNCIÓN_MQ_LONGITUD	4	X'00000004'
MQ_GROUP_ID_LENGTH	24	X'00000018'
MQ_LDAP_PASSWORD_LENGTH	32	X'00000020'
MQ_LISTENER_NAME_LENGTH	48	X'00000030'
MQ_LISTENER_DESC_LENGTH	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH	8	X'00000008'
MQ_LU_NAME_LENGTH	8	X'00000008'
MQ_LUWID_LENGTH	16	X'00000010'
MQ_MAX_EXIT_NAME_LENGTH	128	X'00000080'
MQ_MAX_MCA_USER_ID_LENGTH	64	X'00000040'
MQ_MAX_PROPERTY_NAME_LENGTH	4095	X'00000FFF'
MQ_MAX_USER_ID_LENGTH	64	X'00000040'
MQ_MCA_NOMBRE_TRABAJO_LONGITUD	28	X'0000001C'
MQ_MCA_NAME_LENGTH	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'
MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	(value differs by platform or version)
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
MQ_MSG_ID_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'
MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
LONGITUD_CONTRASEÑA_MQ	12	X'0000000C'
MQ_PROCESS_APPL_ID_LENGTH	256	X'00000100'
MQ_PROCESS_DESC_LENGTH	64	X'00000040'
MQ_PROCESS_ENV_DATA_LENGTH	128	X'00000080'

Tabla 7. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_PROCESS_NAME_LENGTH	48	X'00000030'
MQ_PROCESS_USER_DATA_LENGTH	128	X'00000080'
MQ_NOMBRE_PROGRAMA_LENGTH	20	X'00000014'
MQ_PUT_APPL_NAME_LENGTH	28	X'0000001C'
MQ_PUT_DATE_LENGTH	8	X'00000008'
MQ_PUT_TIME_LENGTH	8	X'00000008'
MQ_Q_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_IDENTIFIER_LENGTH	48	X'00000030'
MQ_Q_MGR_NAME_LENGTH	48	X'00000030'
MQ_Q_NAME_LENGTH	48	X'00000030'
MQ_QSG_NAME_LENGTH	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
MQ_SELECTOR_LENGTH	10240	X'00002800'
MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_LONGITUD_MANDATO	255	X'000000FF'
MQ_SERVICE_DESC_LENGTH	64	X'00000040'
MQ_SERVICE_NAME_LENGTH	32	X'00000020'
MQ_SERVICE_PATH_LENGTH	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH	8	X'00000008'
MQ_SHORT_CONN_NAME_LENGTH	20	X'00000014'
MQ_SHORT_DNAME_LENGTH	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LENGTH	256	X'00000100'
MQ_SSL_HANDSHAKE_STAGE_LENGTH	32	X'00000020'
MQ_SSL_KEY_LIBRARY_LENGTH	44	X'0000002C'
MQ_SSL_KEY_MEMBER_LENGTH	8	X'00000008'
MQ_SSL_KEY_REPOSITORY_LENGTH	256	X'00000100'
MQ_SSL_PEER_NAME_LENGTH	1024	X'00000400'
MQ_SSL_SHORT_PEER_NAME_LENGTH	256	X'00000100'
MQ_START_CODE_LENGTH	4	X'00000004'
MQ_STORAGE_CLASS_DESC_LENGTH	64	X'00000040'
MQ_STORAGE_CLASS_LENGTH	8	X'00000008'
MQ_SUB_IDENTITY_LENGTH	128	X'00000080'
LONGITUD_PUNTO_SUB_MQ	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_B_NONE	1	X'00000001'

Tabla 7. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_SUITE_B_NOT_AVAILABLE	0	X'00000000'
MQ_TCP_NAME_LENGTH	8	X'00000008'
MQ_TIME_LENGTH	8	X'00000008'
MQ_TOPIC_DESC_LENGTH	64	X'00000040'
MQ_NOMBRE_TEMA_LONGITUD	48	X'00000030'
MQ_TOPIC_STR_LENGTH	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	999	X'000003E7'
MQ_TP_NAME_LENGTH	64	X'00000040'
MQ_TPIPE_NAME_LENGTH	8	X'00000008'
MQ_TRAN_INSTANCE_ID_LENGTH	16	X'00000010'
MQ_TRANSACTION_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH	64	X'00000040'
MQ_TRIGGER_PROGRAM_NAME_LENGTH	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH	4	X'00000004'
MQ_USER_ID_LENGTH	12	X'0000000C'
MQ_LONGITUD_VERSIÓN	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

*MQ_ * (Formato de mandato, longitudes de serie)*

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

Tabla 8. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_NOMBRE_ORIGIN_LENGTH	8	X'00000008'
MQ_PSB_NAME_LENGTH	8	X'00000008'
MQ_PST_ID_LENGTH	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH	4	X'00000004'
MQ_RESPONSE_ID_LENGTH	24	X'00000018'
LONGITUD_RBA_MQ	16	X'00000010'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
MQ_UOW_ID_LENGTH	256	X'00000100'
MQ_USER_DATA_LENGTH	10240	X'00002800'
LONGITUD_VOLSER_MQ	6	X'00000006'

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

Tabla 9. Estructuras de constantes	
Nombre	**Estructura**
MQACH_STRUC_ID	"ACH¬"
MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

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

Tabla 10. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQACH_VERSION_1	1	X'00000001'
VERSIÓN_ACTUAL_MQACH_VERSIÓN	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	(value differs by platform or version)
MQACH_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQACT_ * (Señal de contabilidad)

Tabla 11. Nombres y valores de constantes	
Nombre	**Valor**
MQACT_NONE	X'00...00' (32 nulos)
MQACT_NONE_ARRAY	'\0','\0',... (32 nulos)

MQACT_ * (Formato de mandato, Opciones de acción)

Tabla 12. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG	2	X'00000002'
MQACT_COLLECT_STATISTICS	3	X'00000003'
MQACT_PUBSUB	4	X'00000004'

MQACTP_ * (Acción)

Tabla 13. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQACTP_NEW	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
MQACTP_REPORT	3	X'00000003'

MQACTT_ * (Tipos de señal de contabilidad)

Tabla 14. Valores de constantes	
Nombre	**Valor hexadecimal**
MQACTT_UNKNOWN	X'00'
MQACTT_CICS_LUOW_ID	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_DEFAULT	X'05'
MQACTT_UNIX_NUMERIC_ID	X'06'
MQACTT_OS400_ACCOUNT_TOKEN	X'08'
MQACTT_WINDOWS_DEFAULT	X'09'
MQACTT_NT_SECURITY_ID	X'0B'
MQACTT_USER	X'19'

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

Adoptar nuevas comprobaciones de MCA

Tabla 15. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQADOPT_CHECK_NONE	0	X'00000000'
MQADOPT_CHECK_ALL	1	X'00000001'
MQADOPT_CHECK_Q_MGR_NAME	2	X'00000002'
MQADOPT_CHECK_NET_ADDR	4	X'00000004'

Adoptar nuevos tipos de MCA

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

MQAIR_ * (Estructura de registro de información de autenticación)

Tabla 17. Estructuras de constantes	
Nombre	**Estructura**
MQAIR_ID_ESTRUCTURA	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

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

Tabla 18. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
VERSIÓN_ACTUAL_MQAIR_VERSIÓN	2	X'00000002'

MQAIT_ * (Tipo de información de autenticación)

Tabla 19. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Formato de mandato, Valores de estado asíncrono)

Tabla 20. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAS_NONE	0	X'00000000'
MQAS_INICIADO	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_DETENIDO	3	X'00000003'
MQAS_SUSPENDIDO	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ACTIVO	6	X'00000006'

Tabla 20. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAS_INACTIVE	7	X'00000007'

MQAT_ * (Tipos de aplicación de colocación)

Tabla 21. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAT_DESCONOCIDO	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'
MQAT_OS390	2	X'00000002'
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_WINDOWS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIÁN	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'
MQAT_IMS_BRIDGE	19	X'00000013'
MQAT_XCF	20	X'00000014'
MQAT_CICS_PUENTE	21	X'00000015'
MQAT_NOTES_AGENT	22	X'00000016'
MQAT_TPF	23	X'00000017'
USUARIO_MQ	25	X'00000019'
INTERMEDIARIO	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'

Tabla 21. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAT_LOTE	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'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_ * (Formato de mandato, Valores de autorización)

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

MQAUTHOPT_ * (Formato de mandato, Opciones de autorización)

Tabla 23. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAUTHOPT_ACUMULATIVO	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'

Tabla 23. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAUTHOPT_ENTITY_SET	2	X'00000002'
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLICIT	16	X'00000010'

MQAXC_ * (estructura de contexto de salida de API)

Tabla 24. Estructuras de constantes	
Nombre	**Estructura**
MQAXC_STRUC_ID	"AXC¬"
MQAXC_STRUC_ID_ARRAY	'A','X','C','¬'

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

Tabla 25. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAXC_VERSION_1	1	X'00000001'
MQAXC_VERSIÓN_ACTUAL	1	X'00000001'

MQAXP_ * (estructura de parámetros de salida de API)

Tabla 26. Estructuras de constantes	
Nombre	**Estructura**
MQAXP_ID_ESTRUCTURA	"AXP¬"
MQAXP_STRUC_ID_ARRAY	'A','X','P','¬'

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

Tabla 27. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
VERSIÓN_ACTUAL_MQAXP	2	X'00000002'

MQBA_ * (Selectores de atributos de bytes)

Tabla 28. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBA_PRIMERO	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

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

Tabla 29. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBACF_PRIMERO	7001	X'00001B59'

Tabla 29. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBACF_EVENT_ACCOUNTING_TOKEN	7001	X'00001B59'
MQBACF_EVENT_SECURITY_ID	7002	X'00001B5A'
MQBACF_RESPONSE_SET	7003	X'00001B5B'
MQBACF_ID_RESPUESTA	7004	X'00001B5C'
MQBACF_EXTERNAL_UOW_ID	7005	X'00001B5D'
MQBACF_CONNECTION_ID	7006	X'00001B5E'
MQBACF_GENERIC_CONNECTION_ID	7007	X'00001B5F'
MQBACF_ORIGIN_UOW_ID	7008	X'00001B60'
MQBACF_Q_MGR_UOW_ID	7009	X'00001B61'
MQBACF_ACCOUNTING_TOKEN	7010	X'00001B62'
MQBACF_CORREL_ID	7011	X'00001B63'
MQBACF_GROUP_ID	7012	X'00001B64'
MQBACF_MSG_ID	7013	X'00001B65'
MQBACF_CF_LEID	7014	X'00001B66'
MQBACF_DESTINATION_CORREL_ID	7015	X'00001B67'
MQBACF_SUB_ID	7016	X'00001B68'
MQBACF_LAST_USED	7016	X'00001B68'

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

Tabla 30. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBL_NULL_TERMINATED	-1	X'FFFFFFFF'

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

Estructura de opciones de almacenamiento intermedio a manejador de mensajes

Tabla 31. Estructuras de constantes	
Nombre	**Estructura**
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B','M','H','O'

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

Tabla 32. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

Opciones de manejador de almacenamiento intermedio a mensaje

Tabla 33. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIEs	1	X'00000001'

MQBND_ * (Enlaces predeterminados)

Tabla 34. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_ * (Iniciar opciones y estructura)

Estructura de opciones de inicio

Tabla 35. Estructuras de constantes	
Nombre	**Estructura**
MQBO_STRUC_ID	"BO¬¬"
MQBO_STRUC_ID_ARRAY	'B','O','¬','¬'

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

Tabla 36. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

Opciones de inicio

Tabla 37. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBO_NONE	0	X'00000000'

MQBT_ * (Formato de mandato, Tipos de puente)

Tabla 38. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBT_OTMA	1	X'00000001'

MQCA_ * (Selectores de atributos de caracteres)

Tabla 39. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'

Tabla 39. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

Tabla 39. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

Tabla 39. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Formato de mandato, Tipos de parámetros de caracteres)

Tabla 40. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACF_FIRST	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_FROM_PROCESS_NAME	3003	X'00000BBB'
MQCACF_TO_PROCESS_NAME	3004	X'00000BBC'
MQCACF_FROM_NAMELIST_NAME	3005	X'00000BBD'
MQCACF_TO_NAMELIST_NAME	3006	X'00000BBE'
MQCACF_FROM_CHANNEL_NAME	3007	X'00000BBF'
MQCACF_TO_CHANNEL_NAME	3008	X'00000BC0'
MQCACF_FROM_AUTH_INFO_NAME	3009	X'00000BC1'
MQCACF_TO_AUTH_INFO_NAME	3010	X'00000BC2'
MQCACF_Q_NAMES	3011	X'00000BC3'
MQCACF_PROCESS_NAMES	3012	X'00000BC4'
MQCACF_NAMELIST_NAMES	3013	X'00000BC5'

Tabla 40. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
MQCACF_APPL_NAME	3024	X'00000BD0'
MQCACF_USER_IDENTIFIER	3025	X'00000BD1'
MQCACF_AUX_ERROR_DATA_STR_1	3026	X'00000BD2'
MQCACF_AUX_ERROR_DATA_STR_2	3027	X'00000BD3'
MQCACF_AUX_ERROR_DATA_STR_3	3028	X'00000BD4'
MQCACF_BRIDGE_NAME	3029	X'00000BD5'
MQCACF_STREAM_NAME	3030	X'00000BD6'
MQCACF_TOPIC	3031	X'00000BD7'
MQCACF_PARENT_Q_MGR_NAME	3032	X'00000BD8'
MQCACF_ID_CORREL	3033	X'00000BD9'
MQCACF_PUBLISH_TIMESTAMP	3034	X'00000BDA'
MQCACF_STRING_DATA	3035	X'00000BDB'
MQCACF_SUPPORTED_STREAM_NAME	3036	X'00000BDC'
MQCACF_REG_TOPIC	3037	X'00000BDD'
MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_USER_ID	3039	X'00000BDF'
MQCACF_CHILD_Q_MGR_NAME	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NAME	3043	X'00000BE3'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENT_USER_ID	3045	X'00000BE5'
MQCACF_OBJECT_NAME	3046	X'00000BE6'
MQCACF_EVENT_Q_MGR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NAMES	3048	X'00000BE8'
MQCACF_EVENT_APPL_IDENTITY	3049	X'00000BE9'
MQCACF_EVENT_APPL_NAME	3050	X'00000BEA'
MQCACF_EVENT_APPL_ORIGIN	3051	X'00000BEB'
MQCACF_SUBSCRIPTION_NAME	3052	X'00000BEC'

Tabla 40. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY	3054	X'00000BEE'
MQCACF_REG_SUB_IDENTITY	3055	X'00000BEF'
MQCACF_SUBSCRIPTION_USER_DATA	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
MQCACF_DATA_SET_NAME	3059	X'00000BF3'
MQCACF_UOW_START_DATE	3060	X'00000BF4'
MQCACF_UOW_START_TIME	3061	X'00000BF5'
MQCACF_UOW_LOG_START_DATE	3062	X'00000BF6'
MQCACF_UOW_LOG_START_TIME	3063	X'00000BF7'
MQCACF_UOW_LOG_EXTENT_NAME	3064	X'00000BF8'
MQCACF_PRINCIPAL_ENTITY_NAMES	3065	X'00000BF9'
MQCACF_GROUP_ENTITY_NAMES	3066	X'00000BFA'
MQCACF_AUTH_PROFILE_NAME	3067	X'00000BFB'
MQCACF_XX_ENCODE_CASE_ONE nombre_entidad	3068	X'00000BFC'
MQCACF_SERVICE_COMPONENT	3069	X'00000BFD'
MQCACF_RESPONSE_Q_MGR_NAME	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_RESTART_LOG_EXTENT_NAME	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
VÍA DE ACCESO DE REGISTRO DE MQCACF_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN	3079	X'00000C07'
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NAME	3082	X'00000C0A'
ID de MQCACF_PST_ID	3083	X'00000C0B'
MQCACF_TASK_NUMBER	3084	X'00000C0C'
MQCACF_TRANSACTION_ID	3085	X'00000C0D'
MQCACF_Q_MGR_UOW_ID	3086	X'00000C0E'
MQCACF_ORIGIN_NAME	3088	X'00000C10'
MQCACF_ENV_INFO	3089	X'00000C11'
MQCACF_SECURITY_PROFILE	3090	X'00000C12'
FECHA_CONFIGURACIÓN_MQCACF_DATE	3091	X'00000C13'
MQCACF_CONFIGURATION_TIME	3092	X'00000C14'
MQCACF_FROM_CF_STRUC_NAME	3093	X'00000C15'

Tabla 40. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACF_TO_CF_STRUC_NAME	3094	X'00000C16'
MQCACF_CF_STRUC_NAMES	3095	X'00000C17'
MQCACF_FAIL_DATE	3096	X'00000C18'
MQCACF_FAIL_TIME	3097	X'00000C19'
MQCACF_BACKUP_DATE	3098	X'00000C1A'
MQCACF_BACKUP_TIME	3099	X'00000C1B'
MQCACF_SYSTEM_NAME	3100	X'00000C1C'
MQCACF_CF_STRUC_BACKUP_START	3101	X'00000C1D'
MQCACF_CF_STRUC_BACKUP_END	3102	X'00000C1E'
MQCACF_CF_STRUC_LOG_Q_MGRS	3103	X'00000C1F'
MQCACF_FROM_STORAGE_CLASS	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES	3106	X'00000C22'
MQCACF_DSG_NAME	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
MQCACF_SYSP_CMD_USER_ID	3110	X'00000C26'
MQCACF_SYSP_OTMA_GROUP	3111	X'00000C27'
MQCACF_SYSP_OTMA_MEMBER	3112	X'00000C28'
MQCACF_SYSP_OTMA_DRU_EXIT	3113	X'00000C29'
MQCACF_SYSP_OTMA_TPIPE_PFX	3114	X'00000C2A'
MQCACF_SYSP_ARCHIVE_PFX1	3115	X'00000C2B'
MQCACF_SYSP_ARCHIVE_UNIT1	3116	X'00000C2C'
MQCACF_SYSP_LOG_CORREL_ID	3117	X'00000C2D'
MQCACF_SYSP_UNIT_VOLSER	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_DATE	3120	X'00000C30'
MQCACF_SYSP_Q_MGR_RBA	3121	X'00000C31'
MQCACF_SYSP_LOG_RBA	3122	X'00000C32'
MQCACF_SYSP_SERVICE	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME	3125	X'00000C35'
MQCACF_FROM_SERVICE_NAME	3126	X'00000C36'
MQCACF_TO_SERVICE_NAME	3127	X'00000C37'
MQCACF_LAST_PUT_DATE	3128	X'00000C38'
MQCACF_LAST_PUT_TIME	3129	X'00000C39'
MQCACF_LAST_GET_DATE	3130	X'00000C3A'
MQCACF_LAST_GET_TIME	3131	X'00000C3B'
MQCACF_OPERACIÓN_DATE	3132	X'00000C3C'
MQCACF_OPERACIÓN_TIME	3133	X'00000C3D'

Tabla 40. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACF_ACTIVITY_DESC	3134	X'00000C3E'
MQCACF_APPL_IDENTITY_DATA	3135	X'00000C3F'
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
MQCACF_STRUC_ID	3142	X'00000C46'
MQCACF_VALOR_VALOR_Nombre	3143	X'00000C47'
MQCACF_SERVICE_START_DATE	3144	X'00000C48'
MQCACF_SERVICE_START_TIME	3145	X'00000C49'
MQCACF_SYSP_OFFLINE_RBA	3146	X'00000C4A'
MQCACF_SYSP_ARCHIVE_PFX2	3147	X'00000C4B'
MQCACF_SYSP_ARCHIVE_UNIT2	3148	X'00000C4C'
MQCACF_TO_TOPIC_NAME	3149	X'00000C4D'
MQCACF_FROM_TOPIC_NAME	3150	X'00000C4E'
MQCACF_TOPIC_NAMES	3151	X'00000C4F'
MQCACF_SUB_NAME	3152	X'00000C50'
MQCACF_DESTINATION_Q_MGR	3153	X'00000C51'
DESTINO_MQCACF_DESTINO	3154	X'00000C52'
MQCACF_SUB_USER_ID	3156	X'00000C54'
MQCACF_SUB_USER_DATA	3159	X'00000C57'
MQCACF_SUB_SELECTOR	3160	X'00000C58'
MQCACF_LAST_PUB_DATE	3161	X'00000C59'
MQCACF_LAST_PUB_TIME	3162	X'00000C5A'
MQCACF_FROM_SUB_NAME	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
MQCACF_LAST_MSG_TIME	3167	X'00000C5F'
MQCACF_LAST_MSG_DATE	3168	X'00000C60'
MQCACF_SUBSCRIPTION_POINT	3169	X'00000C61'
FILTRO MQCACF_FILTER	3170	X'00000C62'
MQCACF_NONE	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_LAST_USED	3172	X'00000C64'

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

Tabla 41. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCACH_FIRST	3501	X'00000DAD'
MQCACH_CHANNEL_NAME	3501	X'00000DAD'
MQCACH_DESC	3502	X'00000DAE'
MQCACH_MODE_NAME	3503	X'00000DAF'
MQCACH_TP_NAME	3504	X'00000DB0'
MQCACH_XMIT_Q_NAME	3505	X'00000DB1'
MQCACH_CONNECTION_NAME	3506	X'00000DB2'
MQCACH_MCA_NAME	3507	X'00000DB3'
MQCACH_SEC_EXIT_NAME	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME	3509	X'00000DB5'
MQCACH_SEND_EXIT_NAME	3510	X'00000DB6'
MQCACH_RCV_EXIT_NAME	3511	X'00000DB7'
MQCACH_CHANNEL_NAMES	3512	X'00000DB8'
MQCACH_SEC_EXIT_USER_DATA	3513	X'00000DB9'
MQCACH_MSG_EXIT_USER_DATA	3514	X'00000DBA'
MQCACH_SEND_EXIT_USER_DATA	3515	X'00000DBB'
MQCACH_RCV_EXIT_USER_DATA	3516	X'00000DBC'
MQCACH_ID_USUARIO	3517	X'00000DBD'
CONTRASEÑA_MQCACH_PASSWORD	3518	X'00000DBE'
DIRECCIÓN_LOCAL_MQCACH_LOCAL	3520	X'00000DC0'
MQCACH_LOCAL_NAME	3521	X'00000DC1'
MQCACH_LAST_MSG_TIME	3524	X'00000DC4'
MQCACH_LAST_MSG_DATE	3525	X'00000DC5'
MQCACH_MCA_USER_ID	3527	X'00000DC7'
MQCACH_CHANNEL_START_TIME	3528	X'00000DC8'
MQCACH_CHANNEL_START_DATE	3529	X'00000DC9'
MQCACH_MCA_JOB_NAME	3530	X'00000DCA'
MQCACH_LAST_LUWID	3531	X'00000DCB'
MQCACH_CURRENT_LUWID	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
MQCACH_MR_EXIT_USER_DATA	3535	X'00000DCF'
MQCACH_SSL_CIPHER_SPEC	3544	X'00000DD8'
MQCACH_SSL_PEER_NAME	3545	X'00000DD9'
MQCACH_SSL_HANDSHAKE_STAGE	3546	X'00000DDA'
MQCACH_SSL_SHORT_PEER_NAME	3547	X'00000DDB'
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'

Tabla 41. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACH_SSL_CERT_USER_ID	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NAME	3551	X'00000DDF'
DIRECCIÓN_IP_MQCACH_IP	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
MQCACH_LISTENER_NAME	3554	X'00000DE2'
MQCACH_LISTENER_DESC	3555	X'00000DE3'
MQCACH_LISTENER_START_DATE	3556	X'00000DE4'
MQCACH_LISTENER_START_TIME	3557	X'00000DE5'
MQCACH_SSL_KEY_RESET_DATE	3558	X'00000DE6'
MQCACH_SSL_KEY_RESET_TIME	3559	X'00000DE7'
MQCACH_LAST_USED	3559	X'00000DE7'

MQCADSD_ * (Descriptores ADS de cabecera de información de CICS)

Tabla 42. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCADSD_NONE	0	X'00000000'
MQCADSD_SEND	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_ * (Valores de afinidad de conexión)

Tabla 43. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFERIDO	1	X'00000001'

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

Tabla 44. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCAMO_PRIMERO	2701	X'00000A8D'
MQCAMO_CLOSE_DATE	2701	X'00000A8D'
MQCAMO_CLOSE_TIME	2702	X'00000A8E'
MQCAMO_CONN_DATE	2703	X'00000A8F'
MQCAMO_CONN_TIME	2704	X'00000A90'
MQCAMO_DISC_DATE	2705	X'00000A91'
MQCAMO_DISC_TIME	2706	X'00000A92'
MQCAMO_END_DATE	2707	X'00000A93'
MQCAMO_END_TIME	2708	X'00000A94'

Tabla 44. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCAMO_OPEN_DATE	2709	X'00000A95'
MQCAMO_OPEN_TIME	2710	X'00000A96'
MQCAMO_START_DATE	2711	X'00000A97'
MQCAMO_START_TIME	2712	X'00000A98'
MQCAMO_LAST_USED	2712	X'00000A98'

MQCBC_ * (estructura de constantes MQCBC)

Tabla 45. Estructuras de constantes	
Nombre	**Estructura**
MQCBC_STRUC_ID	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

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

Tabla 46. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBCF_ * (distintivos de constantes MQCBC)

Tabla 47. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBCF_NONE	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

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

Tabla 48. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
MQCBCT_REGISTER_CALL	3	X'00000003'
MQCBCT_DEREGISTER_CALL	4	X'00000004'
MQCBCT_EVENT_CALL	5	X'00000005'
MQCBCT_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED	7	X'00000007'

MQCBD_ * (estructura de constantes MQCBD)

Tabla 49. Estructuras de constantes	
Nombre	**Estructura**
MQCBD_ID_STRUCD	"CBD¬"
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

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

Tabla 50. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCBD_VERSION_1	1	X'00000001'
MQCBD_CURRENT_VERSION	1	X'00000001'

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

Tabla 51. Valores de constantes

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

MQCBO_ * (Opciones de creación de bolsa para mqCreatebolsa)

Tabla 52. Valores de constantes

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

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

Tabla 53. Valores de constantes

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

MQCC_ * (códigos de terminación)

Tabla 54. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCC_OK	0	X'00000000'
MQCC_WARNING	1	X'00000001'
MQCC_FAILED	2	X'00000002'
MQCC_DESCONOCIDO	-1	X'FFFFFFFF'

MQCCSI_ * (Identificadores de juego de caracteres codificados)

Tabla 55. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_PREDETERMINADO	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'
MQCCSI_APPL	-3	X'FFFFFFFD'

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

Tabla 56. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_ * (estructura de definición de canal)

Tabla 57. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
Multi MQCD_VERSION_11	11	X'0000000B'
Multi MQCD_CURRENT_VERSION	11	X'0000000B'

Tabla 57. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
z/OS V 9.1.3 MQCD_VERSION_12	12	X'0000000C'
z/OS V 9.1.3 MQCD_CURRENT_VERSION	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 V 9.1.3 MQCD_LENGTH_12	(value differs by platform or version)	(value differs by platform or version)
MQCD_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

MQCDC_ * (Conversión de datos de canal)

Tabla 58. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION	0	X'00000000'

MQCERT_ * (Tipo de política de validación de certificados)

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

MQCF_ * (Distintivos de capacidad)

Tabla 59. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_ * (Recurso de cabecera de información de CICS)

Tabla 60. Nombres y valores de constantes	
Nombre	**Valor hexadecimal**
MQCFAC_NONE	`X'00...00'` (8 nulos)
MQCFAC_NONE_ARRAY	`'\0','\0',...` (8 nulos)

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

Tabla 61. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

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

Tabla 62. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

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

Tabla 63. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFC_LAST	1	X'00000001'
MQCFC_NOT_LAST	0	X'00000000'

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

Tabla 64. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_ * (estructura de cabecera de formato de mandato)

Tabla 65. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFH_STRUC_LENGTH	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
MQCFH_CURRENT_VERSION	3	X'00000003'

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

Tabla 66. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
LONGITUD_ESTRUCTURA_MQCFIF_LENGTH	20	X'00000014'

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

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

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

Tabla 68. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

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

Tabla 69. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIN_STRUC_LENGTH	16	X'00000010'

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

Tabla 70. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIN64_STRUC_LENGTH	24	X'00000018'

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

Opciones de repositorio de renovación de formato de mandato

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

Formato de mandato Eliminar opciones de colas

Tabla 72. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFO_REMOVE_QUEUES_SÍ	1	X'00000001'

Tabla 72. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_ * (Formato de mandato, operadores de filtro)

Tabla 73. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_MAYOR	4	X'00000004'
MQCFOP_NOT_LESS	6	X'00000006'
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NO_MAYOR	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE	21	X'00000015'
MQCFOP_CONTAINS	10	X'0000000A'
MQCFOP_EXCLUDES	13	X'0000000D'
MQCFOP_CONTAINS_GEN	26	X'0000001A'
MQCFOP_EXCLUDES_GEN	29	X'0000001D'

MQCFR_ * (capacidad de recuperación de CF)

Tabla 74. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFR_SÍ	1	X'00000001'
MQCFR_NO	0	X'00000000'

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

Tabla 75. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

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

Tabla 76. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

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

Tabla 77. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_ * (Formato de mandato, Estado de CF)

Tabla 78. Valores de constantes

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

MQCFT_ * (Formato de mandato, Tipos de estructura)

Tabla 79. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFT_NONE	0	X'00000000'
MQCFT_COMMAND	1	X'00000001'
MQCFT_RESPONSE	2	X'00000002'
MQCFT_INTEGER	3	X'00000003'
MQCFT_STRING	4	X'00000004'
MQCFT_INTEGER_LIST	5	X'00000005'
MQCFT_STRING_LIST	6	X'00000006'
MQCFT_EVENT	7	X'00000007'
MQCFT_USER	8	X'00000008'
MQCFT_BYTE_STRING	9	X'00000009'
MQCFT_TRACE_ROUTE	10	X'0000000A'
MQCFT_REPORT	12	X'0000000C'
MQCFT_INTEGER_FILTER	13	X'0000000D'
MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTER	15	X'0000000F'
MQCFT_COMMAND_XR	16	X'00000010'
MQCFT_XR_MSG	17	X'00000011'
MQCFT_XR_ITEM	18	X'00000012'
MQCFT_XR_SUMMARY	19	X'00000013'
MQCFT_GROUP	20	X'00000014'
MQCFT_STATISTICS	21	X'00000015'

Tabla 79. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFT_CONTABILIDAD	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_ * (Formato de mandato, Tipos CF)

Tabla 80. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_ * (Funciones de cabecera de información de CICS)

Tabla 81. Estructuras de constantes	
Nombre	**Estructura**
MQCFUNC_MQCONN	"CONN"
MQCFUNC_MQGET	"GET¬"
MQCFUNC_MQINQ	"INQ¬"
MQCFUNC_MQOPEN	"OPEN"
MQCFUNC_MQPUT	"PUT¬"
MQCFUNC_MQPUT1	"PUT1"
MQCFUNC_NONE	"¬¬¬¬"
MQCFUNC_MQCONN_ARRAY	'C','O','N','N'
MQCFUNC_MQGET_ARRAY	'G','E','T','¬'
MQCFUNC_MQINQ_ARRAY	'I','N','Q','¬'
MQCFUNC_MQOPEN_ARRAY	'O','P','E','N'
MQCFUNC_MQPUT_ARRAY	'P','U','T','¬'
MQCFUNC_MQPUT1_ARRAY	'P','U','T','1'
MQCFUNC_NONE_ARRAY	'¬','¬','¬','¬'

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

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

Tabla 82. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_ * (Definición automática de canal)

Tabla 83. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHAD_DISABLED	0	X'00000000'

Tabla 83. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_ * (Formato de mandato, Estado dudoso)

Tabla 84. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHIDS_NOT_INDOUBT	0	X'00000000'
MQCHIDS_INDOUBT	1	X'00000001'

MQCHLD_ * (Formato de mandato, Disposiciones de canal)

Tabla 85. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHLD_TODOS	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'
MQCHLD_COMPARTIDO	2	X'00000002'
MQCHLD_PRIVATE	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_* (Formato de mandato, Estado del canal)

Tabla 86. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHS_INACTIVE	0	X'00000000'
MQCHS_BINDING	1	X'00000001'
MQCHS_STARTING	2	X'00000002'
MQCHS_RUNNING	3	X'00000003'
MQCHS_STOPPING	4	X'00000004'
MQCHS_RETRYING	5	X'00000005'
MQCHS_STOPPED	6	X'00000006'
MQCHS_REQUESTING	7	X'00000007'
MQCHS_PAUSED	8	X'00000008'
MQCHS_INITIALIZING	13	X'0000000D'
MQCHS_SWITCHING	14	X'0000000E'

MQCHSH_ * (Formato de mandato, Opciones de reinicio compartido de canal)

Tabla 87. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_YES	1	X'00000001'

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

Tabla 88. Valores de constantes

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

MQCHSSTATE_ * (Formato de mandato, Subestados de canal)

Tabla 89. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHSSTATE_OTHER	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_ENVÍO	200	X'000000C8'
MQCHSSTATE_RECEPCIÓN	300	X'0000012C'
MQCHSSTATE_SERIALIZANDO	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'
MQCHSSTATE_LATIDO	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT	1100	X'0000044C'
MQCHSSTATE_IN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONECTANDO	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRIMIENDO	1800	X'00000708'

MQCHT_ * (Tipos de canal)

Tabla 90. Valores de constantes

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

Tabla 90. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHT_CLUSSDR	9	X'00000009'

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

Tabla 91. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_ * (Identificador de correlación)

Tabla 92. Nombres y valores de constantes	
Nombre	**Valor**
MQCI_NONE	X'00...00' (24 nulos)
MQCI_NONE_ARRAY	'\0','\0',... (24 nulos)
MQCI_SESIÓN_NUEVA	X'414D5121...'
MQCI_NEW_SESSION_ARRAY	'\x41','\x4D','\51','\x21',...

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

Estructura de cabecera de información de CICS

Tabla 93. Estructuras de constantes	
Nombre	**Estructura**
MQCIH_STRUC_ID	"CIH¬"
MQCIH_STRUC_ID_ARRAY	'C','I','H','¬'

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

Tabla 94. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_CURRENT_VERSION	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

Distintivos de cabecera de información de CICS

Tabla 95. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCIH_NONE	0	X'00000000'
MQCIH_PASS_EXPIRATION	1	X'00000001'
MQCIH_UNLIMITED_EXPIRATION	0	X'00000000'

Tabla 95. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCIH_REPLY_WITHOUT_NULLS	2	X'00000002'
MQCIH_REPLY_WITH_NULLS	0	X'00000000'
MQCIH_SYNC_ON_RETURN	4	X'00000004'
MQCIH_NO_SYNC_ON_RETURN	0	X'00000000'

MQCLCT_ * (Tipos de memoria caché de clúster)

Tabla 96. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLCT_STATIC	0	X'00000000'
MQCLCT_DYNAMIC	1	X'00000001'

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

Tabla 97. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

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

Tabla 98. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLRT_RETENIDO	1	X'00000001'

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

Tabla 99. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLT_PROGRAM	1	X'00000001'
TRANSACCIÓN_MQC	2	X'00000002'

MQCLWL_ * (Carga de trabajo de clúster)

Tabla 100. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_ * (Tipo de cola de transmisión de clúster)

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

Tabla 101. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Referencia relacionada

"DefClusterXmitQueueTipo (MQLONG)" en la página 828
El atributo `DefClusterXmitQueueTipo` controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Cambiar gestor de colas

Consultar gestor de colas

Consultar gestor de colas (Respuesta)

"MQINQ-Consultar atributos de objeto" en la página 715
La llamada `MQINQ` devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

MQCMD_ * (Códigos de mandato)

Tabla 102. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
MQCMD_PROCESO DE CAMBIO	3	X'00000003'
MQCMD_PROCESO	4	X'00000004'
MQCMD_XX_ENCODE_CASE_ONE create_proceso	5	X'00000005'
MQCMD_DELETE_PROCESS	6	X'00000006'
MQCMD_INQUIRE_PROCESO	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREATE_Q	11	X'0000000B'
MQCMD_DELETE_Q	12	X'0000000C'
MQCMD_INQUIRE_Q	13	X'0000000D'
MQCMD_REFRESH_Q_MGR	16	X'00000010'
MQCMD_RESET_Q_STATS	17	X'00000011'
MQCMD_INQUIRE_Q_NAMES	18	X'00000012'
MQCMD_INQUIRE_PROCESS_NAMES	19	X'00000013'
MQCMD_INQUIRE_CHANNEL_NAMES	20	X'00000014'
MQCMD_CHANGE_CHANNEL	21	X'00000015'
MQCMD_COPY_CHANNEL	22	X'00000016'
MQCMD_CREATE_CHANNEL	23	X'00000017'
MQCMD_DELETE_CHANNEL	24	X'00000018'
MQCMD_INQUIRE_CHANNEL	25	X'00000019'

Tabla 102. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMD_PING_CHANNEL	26	X'0000001A'
MQCMD_RESET_CHANNEL	27	X'0000001B'
MQCMD_START_CHANNEL	28	X'0000001C'
MQCMD_STOP_CHANNEL	29	X'0000001D'
MQCMD_START_CHANNEL_INIT	30	X'0000001E'
MQCMD_START_CHANNEL_LISTENER	31	X'0000001F'
MQCMD_XX_ENCODE_CASE_ONE nombre_cambio	32	X'00000020'
MQCMD_XX_ENCODE_CASE_ONE nombre_cópid_lista	33	X'00000021'
MQCMD_CREATE_NAMELIST	34	X'00000022'
MQCMD_DELETE_NAMELIST	35	X'00000023'
MQCMD_INQUIRE_NAMELIST	36	X'00000024'
MQCMD_INQUIRE_NAMELIST_NAMES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVE_CHANNEL	39	X'00000027'
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_INQUIRE_Q_STATUS	41	X'00000029'
MQCMD_INQUIRE_CHANNEL_STATUS	42	X'0000002A'
MQCMD_CONFIG_EVENT	43	X'0000002B'
MQCMD_Q_MGR_EVENT	44	X'0000002C'
MQCMD_PERFM_EVENT	45	X'0000002D'
MQCMD_CHANNEL_EVENT	46	X'0000002E'
MQCMD_DELETE_PUBLICATION	60	X'0000003C'
MQCMD_DEREGISTER_PUBLISHER	61	X'0000003D'
MQCMD_DEREGISTER_SUBSCRIBER	62	X'0000003E'
MQCMD_PUBLISH	63	X'0000003F'
MQCMD_REGISTER_PUBLISHER	64	X'00000040'
MQCMD_REGISTER_SUSCRIPTOR	65	X'00000041'
MQCMD_REQUEST_UPDATE	66	X'00000042'
MQCMD_BROKER_INTERNAL	67	X'00000043'
MQCMD_ACTIVITY_MSG	69	X'00000045'
MQCMD_INQUIRE_CLUSTER_Q_MGR	70	X'00000046'
MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
MQCMD_SUSPEND_Q_MGR_CLUSTER	72	X'00000048'
MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_CLUSTER	74	X'0000004A'
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'

Tabla 102. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMD_CREATE_AUTH_INFO	81	X'00000051'
MQCMD_DELETE_AUTH_INFO	82	X'00000052'
MQCMD_INQUIRE_AUTH_INFO	83	X'00000053'
MQCMD_INQUIRE_AUTH_INFO_NAMES	84	X'00000054'
MQCMD_INQUIRE_CONNECTION	85	X'00000055'
MQCMD_STOP_CONNECTION	86	X'00000056'
MQCMD_INQUIRE_AUTH_RECS	87	X'00000057'
MQCMD_INQUIRE_ENTITY_AUTH	88	X'00000058'
MQCMD_DELETE_AUTH_REC	89	X'00000059'
MQCMD_SET_AUTH_REC	90	X'0000005A'
MQCMD_LOGGER_EVENT	91	X'0000005B'
MQCMD_RESET_Q_MGR	92	X'0000005C'
MQCMD_CHANGE_LISTENER	93	X'0000005D'
MQCMD_COPY_LISTENER	94	X'0000005E'
MQCMD_CREATE_LISTENER	95	X'0000005F'
MQCMD_DELETE_LISTENER	96	X'00000060'
MQCMD_INQUIRE_LISTENER	97	X'00000061'
MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_SUCESO_MANDATO	99	X'00000063'
MQCMD_CHANGE_SECURITY	100	X'00000064'
MQCMD_CHANGE_CF_STRUC	101	X'00000065'
MQCMD_CHANGE_STG_CLASS	102	X'00000066'
MQCMD_CHANGE_TRACE	103	X'00000067'
MQCMD_ARCHIVE_LOG	104	X'00000068'
MQCMD_BACKUP_CF_STRUC	105	X'00000069'
MQCMD_CREATE_BUFFER_POOL	106	X'0000006A'
MQCMD_CREATE_PAGE_SET	107	X'0000006B'
MQCMD_CREATE_CF_STRUC	108	X'0000006C'
MQCMD_CREATE_STG_CLASS	109	X'0000006D'
MQCMD_COPY_CF_STRUC	110	X'0000006E'
MQCMD_COPY_STG_CLASS	111	X'0000006F'
MQCMD_DELETE_CF_STRUC	112	X'00000070'
MQCMD_DELETE_STG_CLASS	113	X'00000071'
MQCMD_INQUIRE_ARCHIVE	114	X'00000072'
MQCMD_INQUIRE_CF_STRUC	115	X'00000073'
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'

Tabla 102. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMD_INQUIRE_LOG	120	X'00000078'
MQCMD_INQUIRE_SECURITY	121	X'00000079'
MQCMD_INQUIRE_STG_CLASS	122	X'0000007A'
MQCMD_INQUIRE_SYSTEM	123	X'0000007B'
MQCMD_INQUIRE_THREAD	124	X'0000007C'
MQCMD_INQUIRE_TRACE	125	X'0000007D'
MQCMD_INQUIRE_USAGE	126	X'0000007E'
MQCMD_MOVE_Q	127	X'0000007F'
MQCMD_RECOVER_BSDS	128	X'00000080'
MQCMD_RECOVER_CF_STRUC	129	X'00000081'
MQCMD_RESET_TPIPE	130	X'00000082'
MQCMD_RESOLVE_INDOUBT	131	X'00000083'
MQCMD_RESUME_Q_MGR	132	X'00000084'
MQCMD_REVERIFY_SECURIDAD	133	X'00000085'
MQCMD_SET_ARCHIVE	134	X'00000086'
MQCMD_XX_ENCODE_CASE_CAPS_LOCK_ON registro de sesión	136	X'00000088'
MQCMD_SISTEMA	137	X'00000089'
MQCMD_START_CMD_SERVER	138	X'0000008A'
MQCMD_START_Q_MGR	139	X'0000008B'
MQCMD_START_TRACE	140	X'0000008C'
MQCMD_STOP_CHANNEL_INIT	141	X'0000008D'
MQCMD_STOP_CHANNEL_LISTENER	142	X'0000008E'
MQCMD_STOP_CMD_SERVER	143	X'0000008F'
MQCMD_STOP_Q_MGR	144	X'00000090'
MQCMD_STOP_TRACE	145	X'00000091'
MQCMD_SUSPEND_Q_MGR	146	X'00000092'
MQCMD_INQUIRE_CF_STRUC_NAMES	147	X'00000093'
MQCMD_INQUIRE_STG_CLASS_NAMES	148	X'00000094'
MQCMD_CHANGE_SERVICE	149	X'00000095'
MQCMD_COPY_SERVICE	150	X'00000096'
MQCMD_CREATE_SERVICE	151	X'00000097'
MQCMD_DELETE_SERVICE	152	X'00000098'
MQCMD_INQUIRE_SERVICE	153	X'00000099'
MQCMD_INQUIRE_SERVICE_STATUS	154	X'0000009A'
MQCMD_START_SERVICE	155	X'0000009B'
MQCMD_STOP_SERVICE	156	X'0000009C'
MQCMD_DELETE_BUFFER_POOL	157	X'0000009D'
MQCMD_DELETE_PAGE_SET	158	X'0000009E'
MQCMD_CHANGE_BUFFER_POOL	159	X'0000009F'

Tabla 102. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMD_CHANGE_PAGE_SET	160	X'000000A0'
MQCMD_INQUIRE_Q_MGR_STATUS	161	X'000000A1'
MQCMD_CREATE_LOG	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
MQCMD_STATISTICS_CHANNEL	166	X'000000A6'
MQCMD_ACCOUNTING_MQI	167	X'000000A7'
MQCMD_ACCOUNTING_Q	168	X'000000A8'
MQCMD_INQUIRE_AUTH_SERVICE	169	X'000000A9'
MQCMD_CHANGE_TOPIC	170	X'000000AA'
TEMA_COPIA_MQCMD	171	X'000000AB'
MQCMD_CREATE_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC	173	X'000000AD'
MQCMD_INQUIRE_TOPIC	174	X'000000AE'
MQCMD_INQUIRE_TOPIC_NAMES	175	X'000000AF'
MQCMD_INQUIRE_SUBSCRIPTION	176	X'000000B0'
MQCMD_CREATE_SUBSCRIPTION	177	X'000000B1'
MQCMD_CHANGE_SUBSCRIPTION	178	X'000000B2'
MQCMD_DELETE_SUBSCRIPTION	179	X'000000B3'
MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_INQUIRE_SUB_STATUS	182	X'000000B6'
MQCMD_INQUIRE_TOPIC_STATUS	183	X'000000B7'
MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_INQUIRE_PUBSUB_STATUS	185	X'000000B9'
MQCMD_PURGE_CHANNEL	195	X'000000C3'

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

Tabla 103. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMDI_CMDSCOPE_ACEPTADO	1	X'00000001'
MQCMDI_CMDSCOPE_GENERATED	2	X'00000002'
MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
MQCMDI_COMMAND_ACEPTADO	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_QUEUED	6	X'00000006'
MQCMDI_CHANNEL_INIT_STARTED	7	X'00000007'
MQCMDI_RECOVER_STARTED	11	X'0000000B'
MQCMDI_BACKUP_STARTED	12	X'0000000C'
MQCMDI_RECOVER_COMPLETED	13	X'0000000D'

Tabla 103. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMDI_SEC_TIMER_CERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
MQCMDI_SEC_SIGNOFF_ERROR	17	X'00000011'
MQCMDI_IMS_BRIDGE_SUSPENDED	18	X'00000012'
MQCMDI_DB2_SUSPENDED	19	X'00000013'
MQCMDI_DB2_OBSOLETE_MSGS	20	X'00000014'
MQCMDI_SEC_UPPERCASE	21	X'00000015'
MQCMDI_SEC_MIXEDCASE	22	X'00000016'

MQCMDL_* (Niveles de mandatos)

Tabla 104. Nombres y valores de constantes	
Nombre	**Valor**
MQCMDL_LEVEL_800	800
MQCMDL_LEVEL_801	801
MQCMDL_LEVEL_802	802
MQCMDL_LEVEL_900	900
MQCMDL_LEVEL_901	901
MQCMDL_LEVEL_902	902
MQCMDL_LEVEL_903	903
MQCMDL_LEVEL_904	904
MQCMDL_LEVEL_905	905
MQCMDL_LEVEL_910	910
MQCMDL_LEVEL_912	912
MQCMDL_LEVEL_913	913
MQCMDL_LEVEL_914	914
MQCMDL_LEVEL_915	915

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

Crear estructura de opciones de manejador de mensajes

Tabla 105. Estructuras de constantes	
Nombre	**Estructura**
MQCMHO_STRUC_ID	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C','M','H','O'

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

Tabla 106. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_CURRENT_VERSION	1	X'00000001'

Crear opciones de manejador de mensajes

Tabla 107. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'
MQCMHO_VALIDAR	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_ * (Opciones de conexión y estructura)

Estructura de opciones de conexión

Tabla 108. Estructuras de constantes	
Nombre	**Estructura**
MQCNO_STRUC_ID	"CNO¬"
MQCNO_STRUC_ID_ARRAY	'C','N','O','¬'

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

Tabla 109. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCNO_VERSION_1	1	X'00000001'
MQCNO_VERSION_2	2	X'00000002'
MQCNO_VERSION_3	3	X'00000003'
MQCNO_VERSION_4	4	X'00000004'
MQCNO_VERSION_5	5	X'00000005'
MQCNO_CURRENT_VERSION	5	X'00000005'

Opciones de conexión

Tabla 110. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCNO_STANDARD_BINDING	0	X'00000000'
MQCNO_FASTPATH_BINDING	1	X'00000001'
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	2	X'00000002'
MQCNO_SERIALIZE_CONN_TAG_QSG	4	X'00000004'
MQCNO_RESTRICT_CONN_TAG_Q_MGR	8	X'00000008'
MQCNO_RESTRICT_CONN_TAG_QSG	16	X'00000010'
MQCNO_HANDLE_SHARE_NONE	32	X'00000020'
MQCNO_HANDLE_SHARE_BLOCK	64	X'00000040'
MQCNO_HANDLE_SHARE_NO_BLOCK	128	X'00000080'
MQCNO_SHARED_BINDING	256	X'00000100'
MQCNO_ISOLATED_BINDING	512	X'00000200'
MQCNO_LOCAL_BINDING	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'

Tabla 110. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_ONLY	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT	16777216	X'01000000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_ * (Opciones de cierre)

Tabla 111. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCO_INMEDIATO	0	X'00000000'
MQCO_NONE	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_ * (Longitud de datos de salida de cabecera de información de CICS)

Tabla 112. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_ * (Compresión de canal)

Tabla 113. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCOMPRESS_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'

Tabla 113. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
MQCOMPRESS_SISTEMA	8	X'00000008'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_ * (Identificador de conexión)

Tabla 114. Nombres y valores de constantes	
Nombre	**Valor**
MQCONNID_NONE	X'00...00' (24 nulos)
MQCONNID_NONE_ARRAY	'\0','\0',... (24 nulos)

MQCOPY_ * (Opciones de copia de propiedad)

Tabla 115. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
REENVÍO de MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'
INFORME de MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_ * (Tipos de cola de clúster)

Tabla 116. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTE_Q	3	X'00000003'
MQCQT_Q_MGR_ALIAS	4	X'00000004'

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

Tabla 117. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCRC_OK	0	X'00000000'
MQCRC_CICS_ERROR _EXEC_ERROR	1	X'00000001'
MQCRC_MQ_API_ERROR	2	X'00000002'
ERROR DE MQCRC_BRIDGE_	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'
MQCRC_APPLICATION_ABEND	5	X'00000005'
MQCRC_SECURITY_ERROR	6	X'00000006'
MQCRC_PROGRAM_NOT_AVAILABLE	7	X'00000007'

Tabla 117. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NOT_AVAILABLE	9	X'00000009'

MQCS_ * (estado de consumidor de constantes MQCBC)

Tabla 118. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDIDO	3	X'00000003'
MQCS_DETENIDO	4	X'00000004'

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

Tabla 119. Estructuras de constantes	
Nombre	**Estructura**
MQCSC_START	"S¬¬¬"
MQCSC_STARTDATA	"SD¬¬"
MQCSC_TERMINPUT	"TD¬¬"
MQCSC_NONE	"¬¬¬¬"
MQCSC_START_ARRAY	'S','¬','¬','¬'
MQCSC_STARTDATA_ARRAY	'S','D','¬','¬'
MQCSC_TERMINPUT_ARRAY	'T','D','¬','¬'
MQCSC_NONE_ARRAY	'¬','¬','¬','¬'

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

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

Estructura de parámetros de seguridad de conexión

Tabla 120. Estructuras de constantes	
Nombre	**Estructura**
MQCSP_STRUC_ID	"CSP¬"
MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

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

Tabla 121. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSP_VERSION_1	1	X'00000001'
MQCSP_CURRENT_VERSION	1	X'00000001'

Parámetros de seguridad de conexión Tipos de autenticación

Tabla 122. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'

MQCSRV_ * (Opciones de servidor de mandatos)

Tabla 123. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSRV_CONVERT_NO	0	X'00000000'
MQCSRV_CONVERT_YES	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_YES	1	X'00000001'

MQCT_ * (Etiqueta de conexión de gestor de colas)

Tabla 124. Nombres y valores de constantes	
Nombre	**Valor**
MQCT_NONE	X'00...00' (128 nulos)
MQCT_NONE_ARRAY	'\0','\0',... (128 nulos)

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

Tabla 125. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCTES_NOSYNC	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_BACKOUT	4352	X'00001100'
MQCTES_ENDTASK	65536	X'00010000'

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

Estructura de opciones MQCTL

Tabla 126. Estructuras de constantes	
Nombre	**Estructura**
MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

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

Tabla 127. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCTLO_VERSION_1	1	X'00000001'

Tabla 127. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCTLO_CURRENT_VERSION	1	X'00000001'

Opciones de control de consumidor de MQCTL

Tabla 128. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING	8192	X'00002000'

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

Tabla 129. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCUOWC_ONLY	273	X'00000111'
MQCUOWC_CONTINUE	65536	X'00010000'
MQCUOWC_FIRST	17	X'00000011'
MQCUOWC_MIDDLE	16	X'00000010'
MQCUOWC_LAST	272	X'00000110'
MQCUOWC_COMMIT	256	X'00000100'
MQCUOWC_BACKOUT	4352	X'00001100'

MQCXP_ * (Estructura de parámetros de salida de canal)

Tabla 130. Estructuras de constantes	
Nombre	**Estructura**
MQCXP_STRUC_ID	"CXP¬"
MQCXP_STRUC_ID_ARRAY	'C','X','P','¬'

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

Tabla 131. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_VERSION_9	9	X'00000009'
MQCXP_CURRENT_VERSION	9	X'00000009'

MQDC_ * (Clase de destino)

Tabla 132. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDC_GESTIONADO	1	X'00000001'
MQDC_PROPORCIONADO	2	X'00000002'

MQDCC_ * (Opciones de conversión y máscaras y factores)

Opciones de conversión

Tabla 133. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDCC_DEFAULT_CONVERSION	1	X'00000001'
MQDCC_FILL_TARGET_BUFFER	2	X'00000002'
MQDCC_INT_DEFAULT_CONVERSION	4	X'00000004'
MQDCC_SOURCE_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_SOURCE_ENC_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSE	32	X'00000020'
MQDCC_SOURCE_ENC_UNDEFINED	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_TARGET_ENC_NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSE	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'
MQDCC_NONE	0	X'00000000'

Máscaras y factores de opciones de conversión

Tabla 134. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDCC_SOURCE_ENC_MASK	240	X'000000F0'
MQDCC_TARGET_ENC_MASK	3840	X'00000F00'
MQDCC_SOURCE_ENC_FACTOR	16	X'00000010'
MQDCC_TARGET_ENC_FACTOR	256	X'00000100'

MQDELO_ * (Opciones de supresión de publicación/suscripción)

Tabla 135. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL	4	X'00000004'

MQDH_ * (estructura de cabecera de distribución)

Tabla 136. Estructuras de constantes	
Nombre	**Estructura**
MQDH_STRUC_ID	"DH¬¬"
MQDH_STRUC_ID_ARRAY	'D','H','¬','¬'

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

Tabla 137. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_ * (distintivos de cabecera de distribución)

Tabla 138. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_ * (Formato de mandato, Tipos de desconexión)

Tabla 139. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_ * (Listas de distribución)

Tabla 140. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_ * (estructura de cabecera de mensajes no entregados)

Tabla 141. Estructuras de constantes	
Nombre	**Estructura**
MQDLH_STRUC_ID	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D','L','H','¬'

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

Tabla 142. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDLH_VERSION_1	1	X'00000001'
MQDLH_CURRENT_VERSION	1	X'00000001'

MQDLV_ * (Entrega de mensajes permanentes/no persistentes)

Tabla 143. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_ALL	1	X'00000001'
MQDLV_ALL_DUR	2	X'00000002'
MQDLV_ALL_AVAIL	3	X'00000003'

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

Suprimir estructura de opciones de manejador de mensajes

Tabla 144. Estructuras de constantes	
Nombre	**Estructura**
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D','M','H','O'

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

Tabla 145. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_CURRENT_VERSION	1	X'00000001'

Suprimir opciones de manejador de mensajes

Tabla 146. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMHO_NONE	0	X'00000000'

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

Suprimir estructura de opciones de propiedad de mensaje

Tabla 147. Estructuras de constantes	
Nombre	**Estructura**
MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

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

Tabla 148. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_CURRENT_VERSION	1	X'00000001'

Suprimir opciones de propiedad de mensaje

Tabla 149. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_ * (WLM de DNS)

Tabla 150. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_SÍ	1	X'00000001'

MQDT_ * (Tipos de destino)

Tabla 151. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_ * (estructura de parámetros de salida de conversión)

Tabla 152. Estructuras de constantes

Nombre	Estructura
MQDXP_STRUC_ID	"DXP¬"
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

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

Tabla 153. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_ * (Valores de señal)

Tabla 154. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQEC_MSG_LLEGADO	2	X'00000002'
MQEC_WAIT_INTERVAL_CADUCADO	3	X'00000003'
MQEC_WAIT_CANCELADO	4	X'00000004'
MQEC_Q_MGR_QUIESCING	5	X'00000005'
MQEC_CONNECTION_QUIESCING	6	X'00000006'

MQEI_ * (Caducidad)

Tabla 155. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_ * (Codificación)

MQENC_ * (Codificación)

Tabla 156. Valores de constantes por plataforma			
Nombre	**Plataforma**	**Valor decimal**	**Valor hexadecimal**
MQENC_NATIVE	IBM i	273	X'00000111 "
	Linux	546	X'00000222 "
	Linux en SPARC	273	X'00000111 "
	Linux en x86	546	X'00000222 "
	Solaris en SPARC	273	X'00000111 "
	UNIX	273	X'00000111 "
	Windows	546	X'00000222 "
	Micro Focus COBOL en Windows	17	X'00000011 "
	z/OS	785	X'00000311 "

Tabla 157. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MÁSCARA_ENTERO_MQENC_MÁSCARA	15	X'0000000F'
MÁSCARA_DECIMAL_MQENC_MÁSCARA	240	X'000000F0'
Máscara MQENC_FLOAT_MASK	3840	X'00000F00'
MÁSCARA_RESERVA_MQENC_RESERVADO	-4096	X'FFFFF000'

MQENC_ * (Codificaciones para enteros binarios)

Tabla 158. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_ * (Codificaciones para enteros decimales empaquetados)

Tabla 159. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_DECIMAL_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_ * (Codificaciones para números de coma flotante)

Tabla 160. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQENC_FLOAT_TNS	1024	X'00000400'

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

Estructura de cabecera de formato de mandato incorporado

Tabla 161. Estructuras de constantes	
Nombre	**Estructura**
MQEPH_STRUC_ID	"EPH¬"
MQEPH_STRUC_ID_ARRAY	'E','P','H','¬'

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

Tabla 162. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_CURRENT_VERSION	1	X'00000001'

Distintivos de cabecera de formato de mandato incorporado

Tabla 163. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_ * (Formato de mandato, Tipos de escape)

Tabla 164. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQET_MQSC	1	X'00000001'

MQEVO_ * (Formato de mandato, Orígenes de sucesos)

Tabla 165. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEVO_OTHER	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT	2	X'00000002'

Tabla 165. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'
MQEVO_MQSUB	6	X'00000006'
MQEVO_CTLMSG	7	X'00000007'
V 9.1.0 V 9.1.0 MQEVO_REST	8	X'00000008'

MQEVR_ * (Formato de mandato, Registro de sucesos)

Tabla 166. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPCIÓN	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_ * (Intervalo de exploración de caducidad)

Tabla 167. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEXPI_OFF	0	X'00000000'

MQFB_ * (Valores de comentarios)

Tabla 168. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFB_NONE	0	X'00000000'
MQFB_SISTEMA_PRIMERO	1	X'00000001'
MQFB_QUIT	256	X'00000100'
MQFB_EXPIRATION	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
MQFB_CHANNEL_COMPLETED	262	X'00000106'
MQFB_CHANNEL_FAIL_RETRY	263	X'00000107'
MQFB_CHANNEL_FAIL	264	X'00000108'
MQFB_APPL_CANNOT_BE_STARTED	265	X'00000109'
MQFB_TM_ERROR	266	X'0000010A'
MQFB_APPL_TYPE_ERROR	267	X'0000010B'
MQFB_STOPPED_BY_MSG_EXIT	268	X'0000010C'
MQFB_ACTIVITY	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
MQFB_PAN	275	X'00000113'

Tabla 168. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFB_NAN	276	X'00000114'
MQFB_STOPPED_BY_CHAD_EXIT	277	X'00000115'
MQFB_STOPPED_BY_PUBSUB_EXIT	279	X'00000117'
MQFB_NOT_A_REPOSITORY_MSG	280	X'00000118'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MAX_ACTIVITIES	282	X'0000011A'
MQFB_NO_REENVIADO	283	X'0000011B'
MQFB_NO_ENTREGADO	284	X'0000011C'
MQFB_UNSUPPORTED_FORWARDING	285	X'0000011D'
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
MQFB_DATA_LENGTH_ZERO	291	X'00000123'
MQFB_DATA_LENGTH_NEGATIVO	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
MQFB_IIH_ERROR	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_ERROR DE INTERNAL	401	X'00000191'
MQFB_CICS_NO_AUTORIZADO	402	X'00000192'
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
MQFB_CICS_ID_CORREL_ERROR	404	X'00000194'
MQFB_CICS_ERROR _CCSID	405	X'00000195'
MQFB_CICS_ERROR _ENCODING_	406	X'00000196'
MQFB_CICS_ERROR _CIH	407	X'00000197'
MQFB_CICS_ERROR DE UOW	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_ERROR _DLQ	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIBER_IS_PUBLISHER	502	X'000001F6'
MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'

Tabla 168. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFB_SYSTEM_LAST	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_ * (Formato de mandato, Forzar opciones)

Tabla 169. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFC_YES	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_ * (Formatos)

Tabla 170. Nombres y valores de constantes	
Nombre	**Valor**
MQFMT_NONE	"¬¬¬¬¬¬¬¬"
MQFMT_ADMIN	"MQADMIN¬"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM¬"
MQFMT_CICS	"MQCICS¬¬"
MQFMT_COMMAND_1	"MQCMD1¬¬"
MQFMT_COMMAND_2	"MQCMD2¬¬"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD¬¬"
MQFMT_DIST_HEADER	"MQHDIST¬"
MQFMT_EMBEDDED_PCF	"MQHEPCF¬"
MQFMT_EVENT	"MQEVENT¬"
MQFMT_IMS	"MQIMS¬¬¬"
MQFMT_IMS_VAR_STRING	"MQIMSVS¬"
MQFMT_MD_EXTENSION	"MQHMDE¬¬"
MQFMT_PCF	"MQPCF¬¬¬"
MQFMT_REF_MSG_HEADER	"MQHREF¬¬"
MQFMT_RF_HEADER	"MQHRF¬¬¬"
MQFMT_RF_HEADER_1	"MQHRF¬¬¬"
MQFMT_RF_HEADER_2	"MQHRF2¬¬"
MQFMT_STRING	"MQSTR¬¬¬"
MQFMT_TRIGGER	"MQTRIG¬¬"
MQFMT_WORK_INFO_HEADER	"MQHWIH¬¬"
MQFMT_XMIT_Q_HEADER	"MQXMIT¬¬"
MQFMT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'
MQFMT_ADMIN_ARRAY	'M','Q','A','D','M','I','N','¬'
MQFMT_CHANNEL_COMPLETED_ARRAY	'M','Q','C','H','C','O','M','¬'
MQFMT_CICS_ARRAY	'M','Q','C','I','C','S','¬','¬'

Tabla 170. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQFMT_COMMAND_1_ARRAY	'M','Q','C','M','D','1','¬','¬'
MQFMT_COMMAND_2_ARRAY	'M','Q','C','M','D','2','¬','¬'
MQFMT_DEAD_LETTER_HEADER_ARRAY	'M','Q','D','E','A','D','¬','¬'
MQFMT_DIST_HEADER_ARRAY	'M','Q','H','D','I','S','T','¬'
MQFMT_EMBEDDED_PCF_ARRAY	'M','Q','H','E','P','C','F','¬'
MQFMT_EVENT_ARRAY	'M','Q','E','V','E','N','T','¬'
MQFMT_IMS_ARRAY	'M','Q','I','M','S','¬','¬','¬'
MQFMT_IMS_VAR_STRING_ARRAY	'M','Q','I','M','S','V','S','¬'
MQFMT_MD_EXTENSION_ARRAY	'M','Q','H','M','D','E','¬','¬'
MQFMT_PCF_ARRAY	'M','Q','P','C','F','¬','¬','¬'
MQFMT_REF_MSG_HEADER_ARRAY	'M','Q','H','R','E','F','¬','¬'
MQFMT_RF_HEADER_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_1_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_2_ARRAY	'M','Q','H','R','F','2','¬','¬'
MQFMT_STRING_ARRAY	'M','Q','S','T','R','¬','¬','¬'
MQFMT_TRIGGER_ARRAY	'M','Q','T','R','I','G','¬','¬'
MQFMT_WORK_INFO_HEADER_ARRAY	'M','Q','H','W','I','H','¬','¬'
MQFMT_XMIT_Q_HEADER_ARRAY	'M','Q','X','M','I','T','¬','¬'

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

MQFUN_ * (Tipos de función de aplicación)

Tabla 171. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFUN_TYPE_UNKNOWN	0	X'00000000'
JVM MQFUN_TYPE_JVM	1	X'00000001'
PROGRAMA_TIPO_MQFUN_PROGRAMA	2	X'00000002'
MQFUN_TIPO_PROCEDIMIENTO	3	X'00000003'
MQFUN_TYPE_USERDEF	4	X'00000004'
MQFUN_TYPE_COMMAND	5	X'00000005'

MQGA_ * (Selectores de atributos de grupo)

Tabla 172. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGA_PRIMERO	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

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

Tabla 173. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGACF_PRIMERO	8001	X'00001F41'
CONTEXTO_MANDATO_MQGACF_MANDATO	8001	X'00001F41'
MQGACF_COMMAND_DATOS	8002	X'00001F42'
MQGACF_TRACE_ROUTE	8003	X'00001F43'
MQGACF_OPERACIÓN	8004	X'00001F44'
MQGACF_ACTIVIDAD	8005	X'00001F45'
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MQGACF_MENSAJE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_VALOR_NAMING	8009	X'00001F49'
MQGACF_Q_ACCOUNTING_DATA	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_USED	8012	X'00001F4C'

MQGI_ * (Identificador de grupo)

Tabla 174. Nombres y valores de constantes	
Nombre	**Valor**
MQGI_NONE	X'00...00' (24 nulos)
MQGI_NONE_ARRAY	'\0','\0',... (24 nulos)

MQGMO_ * (Obtener opciones de mensaje y estructura)

Estructura de opciones de obtención de mensajes

Tabla 175. Estructuras de constantes	
Nombre	**Estructura**
MQGMO_STRUC_ID	"GMO¬"
MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

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

Tabla 176. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

Obtener opciones de mensaje

Tabla 177. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGMO_WAIT	1	X'00000001'
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF_QUIESCING	8192	X'00002000'
MQGMO_SYNCPOINT	2	X'00000002'
MQGMO_SYNCPOINT_IF_PERSISTENT	4096	X'00001000'
MQGMO_NO_SYNCPOINT	4	X'00000004'
MQGMO_MARK_SKIP_BACKOUT	128	X'00000080'
MQGMO_BROWSE_FIRST	16	X'00000010'
MQGMO_BROWSE_NEXT	32	X'00000020'
MQGMO_BROWSE_MSG_UNDER_CURSOR	2048	X'00000800'
MQGMO_BROWSE_HANDLE	17825808	X'01100010'
MQGMO_BROWSE_CO_OP	18874384	X'01200010'
MQGMO_MSG_UNDER_CURSOR	256	X'00000100'
MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_ * (Estado de grupo)

Tabla 178. Nombres y valores de constantes	
Nombre	**Valor**
MQGS_NOT_IN_GROUP	'¬'

Tabla 178. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

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

MQHA_ * (Selectores de descriptor de contexto)

Tabla 179. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHA_PRIMERO	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_ * (Descriptores de paquete)

Tabla 180. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_ * (Descriptores de contexto de conexión)

Tabla 181. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_ * (Descriptor de mensaje)

Tabla 182. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_ * (manejador de objetos)

Tabla 183. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_ * (Formato de mandato, Estados de manejador)

Tabla 184. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHSTATE_INACTIVE	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_ * (Selectores de atributos enteros)

Tabla 185. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPT_CONTEXT	260	X'00000104'
V 9.1.0 MQ Adv. VUE V 9.1.0 MQ Adv. VUE MQIA_ADVANCED_CAPABILITY	273	X'00000111'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
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'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_FIRST	1	X'00000001'
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'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_MONITOR_INTERVAL	81	X'00000051'
MQIA_MONITORING_AUTO_CLUSSDR	124	X'0000007C'
MQIA_MONITORING_CHANNEL	122	X'0000007A'
MQIA_MONITORING_Q	123	X'0000007B'
MQIA_MSG_DELIVERY_SEQUENCE	16	X'00000010'
MQIA_MSG_DEQ_COUNT	38	X'00000026'
MQIA_MSG_ENQ_COUNT	37	X'00000025'
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'
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'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_QMGR_CFCONLOS	245	X'000000F5'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'
MQIA_QMOPT_CONS_ERROR_MSGS	153	X'00000099'
MQIA_QMOPT_CONS_INFO_MSGS	151	X'00000097'
MQIA_QMOPT_CONS_REORG_MSGS	156	X'0000009C'
MQIA_QMOPT_CONS_SYSTEM_MSGS	157	X'0000009D'
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'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'
MQIA_STATISTICS_INTERVAL	131	X'00000083'
MQIA_STATISTICS_MQI	127	X'0000007F'
MQIA_STATISTICS_Q	128	X'00000080'
MQIA_SUB_COUNT	204	X'000000CC'
MQIA_SUB_SCOPE	218	X'000000DA'
MQIA_SYNCPOINT	30	X'0000001E'
MQIA_TCP_CHANNELS	114	X'00000072'
MQIA_TCP_KEEP_ALIVE	115	X'00000073'
MQIA_TCP_STACK_TYPE	116	X'00000074'
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_ * (Formato de mandato, Tipos de parámetros enteros)

Tabla 186. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_FIRST	1001	X'000003E9'
MQIACF_Q_MGR_ATTRS	1001	X'000003E9'
MQIACF_Q_ATTRS	1002	X'000003EA'
MQIACF_PROCESS_ATTRS	1003	X'000003EB'
MQIACF_NAMELIST_ATTRS	1004	X'000003EC'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_FORCE	1005	X'000003ED'
MQIACF_REPLACE	1006	X'000003EE'
MQIACF_PURGE	1007	X'000003EF'
MQIACF_QUIESCE	1008	X'000003F0'
MQIACF_MODE	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'
MQIACF_EVENT_APPL_TYPE	1010	X'000003F2'
MQIACF_EVENT_ORIGIN	1011	X'000003F3'
MQIACF_PARAMETER_ID	1012	X'000003F4'
MQIACF_ERROR_ID	1013	X'000003F5'
MQIACF_ERROR_IDENTIFIER	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
MQIACF_OBJECT_TYPE	1016	X'000003F8'
MQIACF_TIPO_ESCAPE_ES	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
MQIACF_REASON_QUALIFIER	1020	X'000003FC'
MQIACF_COMMAND	1021	X'000003FD'
MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_OPEN_TYPE	1023	X'000003FF'
MQIACF_PROCESS_ID	1024	X'00000400'
MQIACF_THREAD_ID	1025	X'00000401'
MQIACF_Q_STATUS_ATTRS	1026	X'00000402'
MQIACF_UNCOMMITTED_MSGS	1027	X'00000403'
MQIACF_HANDLE_ESTADO	1028	X'00000404'
MQIACF_AUX_ERROR_DATA_INT_1	1070	X'0000042E'
MQIACF_AUX_ERROR_DATA_INT_2	1071	X'0000042F'
MQIACF_CONV_REASON_CODE	1072	X'00000430'
MQIACF_BRIDGE_TYPE	1073	X'00000431'
MQIACF_CONSULTA	1074	X'00000432'
MQIACF_INTERVALO_ESPERA	1075	X'00000433'
MQIACF_OPCIONES	1076	X'00000434'
MQIACF_BROKER_OPCIONES	1077	X'00000435'
MQIACF_REFRESH_TYPE	1078	X'00000436'
MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
MQIACF_DATOS	1080	X'00000438'
MQIACF_REGISTRATION_OPTIONS	1081	X'00000439'
MQIACF_PUBCIONTION_OPCIONES	1082	X'0000043A'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_CLUSTER_INFO	1083	X'0000043B'
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
MQIACF_Q_MGR_TYPE	1085	X'0000043D'
MQIACF_ACCIÓN	1086	X'0000043E'
MQIACF_SUSPEND	1087	X'0000043F'
MQIACF_BROKER_COUNT	1088	X'00000440'
MQIACF_APPL_COUNT	1089	X'00000441'
MQIACF_ANONOUS_COUNT	1090	X'00000442'
MQIACF_REG_REG_OPCIONES	1091	X'00000443'
MQIACF_DELETE_OPCIONES	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
MQIACF_REFRESH_INTERVAL	1094	X'00000446'
MQIACF_REFRESH_REPOSITORY	1095	X'00000447'
MQIACF_REMOVE_QUEUES	1096	X'00000448'
MQIACF_OPEN_INPUT_TYPE	1098	X'0000044A'
MQIACF_OPEN_OUTPUT	1099	X'0000044B'
MQIACF_OPEN_SET	1100	X'0000044C'
MQIACF_OPEN_INQUIRE	1101	X'0000044D'
MQIACF_OPEN_BROWSE	1102	X'0000044E'
MQIACF_Q_STATUS_TYPE	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
MQIACF_Q_STATUS	1105	X'00000451'
MQIACF_SECURITY_TYPE	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
MQIACF_CONNECT_OPTIONS	1108	X'00000454'
MQIACF_CONN_INFO_TYPE	1110	X'00000456'
MQIACF_CONN_INFO_CONN	1111	X'00000457'
MQIACF_CONN_INFO_HANDLE	1112	X'00000458'
MQIACF_CONN_INFO_ALL	1113	X'00000459'
MQIACF_AUTH_PROFILE_ATTRS	1114	X'0000045A'
MQIACF_AUTORIZATION_LIST	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_tipo_entidad	1118	X'0000045E'
MQIACF_COMMAND_INFO	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SYSTEM	1122	X'00000462'
MQIACF_Q_MGR_EVENT	1123	X'00000463'
MQIACF_Q_MGR_DQM	1124	X'00000464'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
MQIACF_QSG_DISPS	1126	X'00000466'
MQIACF_UOW_ESTADO	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
MQIACF_CF_STRUC_STATUS	1130	X'0000046A'
MQIACF_TIPO_UOW	1132	X'0000046C'
MQIACF_CF_STRUC_ATTRS	1133	X'0000046D'
MQIACF_EXCLUDE_INTERVAL	1134	X'0000046E'
MQIACF_CF_STATUS_TYPE	1135	X'0000046F'
MQIACF_CF_STATUS_SUMMARY	1136	X'00000470'
MQIACF_CF_STATUS_CONNECT	1137	X'00000471'
MQIACF_CF_STATUS_BACKUP	1138	X'00000472'
MQIACF_CF_STRUC_TYPE	1139	X'00000473'
MQIACF_CF_STRUC_SIZE_MAX	1140	X'00000474'
MQIACF_CF_STRUC_SIZE_USED	1141	X'00000475'
MQIACF_CF_STRUC_ENTRIES_MAX	1142	X'00000476'
MQIACF_CF_STRUC_ENTRIES_USED	1143	X'00000477'
MQIACF_CF_STRUC_BACKUP_SIZE	1144	X'00000478'
Tipo_MOVE_MQIACF_MQ	1145	X'00000479'
MQIACF_MOVE_TYPE_MOVE	1146	X'0000047A'
MQIACF_MOVE_TYPE_ADD	1147	X'0000047B'
MQIACF_Q_MGR_NUMBER	1148	X'0000047C'
MQIACF_Q_MGR_ESTADO	1149	X'0000047D'
MQIACF_Db2_ESTADO	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_SECURITY_INTERVAL	1153	X'00000481'
MQIACF_SECURITY_SWITCH	1154	X'00000482'
MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
tipo_USAG_MQIACF	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
MQIACF_USAGE_PERSIST_PAGES	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
MQIACF_PAGESET_STATUS	1165	X'0000048D'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
MQIACF_USAGE_DATA_SET_TYPE	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'
MQIACF_MOVE_COUNT	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURATION_OBJECTS	1173	X'00000495'
MQIACF_CONFIGURATION_EVENTS	1174	X'00000496'
MQIACF_SYSP_TYPE	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVE	1177	X'00000499'
MQIACF_SYSP_MAX_READ_TAPES	1178	X'0000049A'
MQIACF_SYSP_IN_BUFFER_SIZE	1179	X'0000049B'
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
MQIACF_SYSP_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVE	1183	X'0000049F'
MQIACF_SYSP_DUAL_ARCHIVE	1184	X'000004A0'
MQIACF_SYSP_DUAL_BSDS	1185	X'000004A1'
MQIACF_SYSP_MAX_CONNS	1186	X'000004A2'
MQIACF_SYSP_MAX_CONNS_FORE	1187	X'000004A3'
MQIACF_SYSP_MAX_CONNS_BACK	1188	X'000004A4'
MQIACF_SYSP_EXIT_INTERVAL	1189	X'000004A5'
MQIACF_SYSP_EXIT_TASKS	1190	X'000004A6'
MQIACF_SYSP_CHKPOINT_COUNT	1191	X'000004A7'
MQIACF_SYSP_OTMA_INTERVAL	1192	X'000004A8'
MQIACF_SYSP_Q_INDEX_DEFER	1193	X'000004A9'
MQIACF_SYSP_Db2_TAREAS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_SYSP_ARCHIVE_WTOR	1205	X'000004B5'
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'
MQIACF_SYSP_ALLOC_SECONDARY	1210	X'000004BA'
MQIACF_SYSP_PROTECT	1211	X'000004BB'
MQIACF_SYSP_QUIESCE_INTERVAL	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
MQIACF_SYSP_UNIT_ADDRESS	1214	X'000004BE'
MQIACF_SYSP_UNIT_STATUS	1215	X'000004BF'
MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
MQIACF_SYSP_OFFLOAD_STATUS	1219	X'000004C3'
MQIACF_SYSP_TOTAL_LOGS	1220	X'000004C4'
MQIACF_SYSP_FULL_LOGS	1221	X'000004C5'
MQIACF_LISTENER_ATTRS	1222	X'000004C6'
MQIACF_LISTENER_STATUS_ATTRS	1223	X'000004C7'
MQIACF_SERVICE_ATTRS	1224	X'000004C8'
MQIACF_SERVICE_STATUS_ATTRS	1225	X'000004C9'
MQIACF_Q_TIEMPO_INDICADOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
MQIACF_AUTO_OPCIONES	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
MQIACF_CONNECTION_COUNT	1230	X'000004CE'
MQIACF_Q_MGR_FACILITY	1231	X'000004CF'
MQIACF_CHINIT_STATUS	1232	X'000004D0'
MQIACF_CMD_SERVER_STATUS	1233	X'000004D1'
MQIACF_ROUTE_DETAIL	1234	X'000004D2'
MQIACF_RECORDED_ACTIVITIES	1235	X'000004D3'
MQIACF_MAX_ACTIVITIES	1236	X'000004D4'
MQIACF_DISCONTINUITY_COUNT	1237	X'000004D5'
MQIACF_ROUTE_ACUMULACIÓN_RUTA	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
MQIACF_OPERACIÓN_TYPE	1240	X'000004D8'
MQIACF_BACKOUT_COUNT	1241	X'000004D9'
MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_EXPIRY	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'
MQIACF_ORIGINAL_LENGTH	1251	X'000004E3'
MQIACF_PERSISTENCE	1252	X'000004E4'
MQIACF_PRIORITY	1253	X'000004E5'
MQIACF_REASON_CODE	1254	X'000004E6'
MQIACF_REPORT	1255	X'000004E7'
MQIACF_VERSION	1256	X'000004E8'
MQIACF_UNRECORDED_ACTIVITIES	1257	X'000004E9'
MQIACF_SUPERVISIÓN	1258	X'000004EA'
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
MQIACF_SERVICE_STATUS	1260	X'000004EC'
MQIACF_Q_TYPES	1261	X'000004ED'
MQIACF_USER_ID_SUPPORT	1262	X'000004EE'
MQIACF_INTERFACE_VERSION	1263	X'000004EF'
MQIACF_AUTH_SERVICE_ATTRS	1264	X'000004F0'
MQIACF_USAGE_EXPAND_TYPE	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_Db2_BLOB_TASKS	1267	X'000004F3'
MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
MQIACF_TOPIC_ATTRS	1269	X'000004F5'
MQIACF_PUBSUB_PROPERTIEs	1271	X'000004F7'
MQIACF_DESTINATION_CLASS	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SUBSCRIPTION_SCOPE	1275	X'000004FB'
MQIACF_VARIABLE_USER_ID	1277	X'000004FD'
MQIACF_REQUEST_ONLY	1280	X'00000500'
MQIACF_PUB_PRIORITY	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
MQIACF_WILDCARD_SCHEMA	1288	X'00000508'
MQIACF_SUB_TYPE	1289	X'00000509'
MQIACF_MESSAGE_COUNT	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSION	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_TOPIC_STATUS	1295	X'0000050F'
MQIACF_TOPIC_SUB	1296	X'00000510'
MQIACF_TOPIC_PUB	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_TOPIC_STATUS_ATTRS	1301	X'00000515'
MQIACF_TOPIC_STATUS_TYPE	1302	X'00000516'
MQIACF_SUB_OPTIONS	1303	X'00000517'
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_TIPO_CLEAR	1305	X'00000519'
MQIACF_CLEAR_SCOPE	1306	X'0000051A'
MQIACF_SUB_LEVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE	1308	X'0000051C'
MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
Tipo_selección_MQIACF	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_CHLAUTH_TYPE	1352	X'00000548'
MQXR_DIAGNOSTICS_TYPE	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
MQIACF_OPERATION_ID	1356	X'0000054C'
MQIACF_API_CALLER_TYPE	1357	X'0000054D'
MQIACF_API_ENVIRONMENT	1358	X'0000054E'
MQIACF_TRACE_DETAIL	1359	X'0000054F'
MQIACF_HOBJ	1360	X'00000550'
MQIACF_CALL_TYPE	1361	X'00000551'
MQIACF_MQCB_OPERATION	1362	X'00000552'
MQIACF_MQCB_TYPE	1363	X'00000553'
MQIACF_MQCB_OPTIONS	1364	X'00000554'
MQIACF_CLOSE_OPTIONS	1365	X'00000555'
MQIACF_CTL_OPERATION	1366	X'00000556'
MQIACF_GET_OPTIONS	1367	X'00000557'
MQIACF_RECS_PRESENT	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'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_PUT_OPTIONS	1373	X'0000055D'
MQIACF_BUFFER_LENGTH	1374	X'0000055E'
MQIACF_TRACE_DATA_LENGTH	1375	X'0000055F'
MQIACF_SMDS_EXPANDST	1376	X'00000560'
MQIACF_STRUC_LENGTH	1377	X'00000561'
MQIACF_ITEM_COUNT	1378	X'00000562'
MQIACF_EXPIRY_TIME	1379	X'00000563'
MQIACF_CONNECT_TIME	1380	X'00000564'
MQIACF_DISCONNECT_TIME	1381	X'00000565'
MQIACF_HSUB	1382	X'00000566'
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	1391	X'0000056F'
MQIACF_SELECTORS	1392	X'00000570'
MQIACF_INTATTR_COUNT	1393	X'00000571'
MQIACF_INTATTRS	1394	X'00000572'
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	1408	X'00000580'
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'

Tabla 186. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_SYSP_MAX_CONC_OFFLOADS	1413	X'00000585'
MQIACF_SYSP_ZHYPERWRITE	1414	X'00000586'
MQIACF_Q_MGR_STATUS_LOG	1415	X'00000587'
MQIACF_ARCHIVE_LOG_SIZE	1416	X'00000588'
MQIACF_MEDIA_LOG_SIZE	1417	X'00000589'
MQIACF_RESTART_LOG_SIZE	1418	X'0000058A'
MQIACF_REUSABLE_LOG_SIZE	1419	X'0000058B'
MQIACF_LOG_IN_USE	1420	X'0000058C'
MQIACF_LOG_UTILIZACIÓN	1421	X'0000058D'
V 9.1.1 V 9.1.1 MQIACF_IGNORE_ESTADO	1423	X'0000058F'
MQIACF_LAST_USED	1423	X'0000058F'

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

Tabla 187. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'
MQIACH_DISC_INTERVAL	1503	X'000005DF'
MQIACH_SHORT_TIMER	1504	X'000005E0'
MQIACH_SHORT_RETRY	1505	X'000005E1'
MQIACH_LONG_TIMER	1506	X'000005E2'
MQIACH_LONG_RETRY	1507	X'000005E3'
MQIACH_PUT_AUTHORITY	1508	X'000005E4'
MQIACH_SEQUENCE_NUMBER_WRAP	1509	X'000005E5'
MQIACH_MAX_MSG_LENGTH	1510	X'000005E6'
MQIACH_CHANNEL_TYPE	1511	X'000005E7'
MQIACH_DATA_COUNT	1512	X'000005E8'
MQIACH_NAME_COUNT	1513	X'000005E9'
MQIACH_MSG_SEQUENCE_NUMBER	1514	X'000005EA'
MQIACH_DATA_CONVERSION	1515	X'000005EB'
MQIACH_IN_DOUBT	1516	X'000005EC'
MQIACH_MCA_TYPE	1517	X'000005ED'
MQIACH_SESSION_COUNT	1518	X'000005EE'
MQIACH_ADAPTER	1519	X'000005EF'
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'

Tabla 187. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'
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'

Tabla 187. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'
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'

Tabla 187. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACH_CURRENT_SHARING_CONVS	1617	X'00000651'
MQIACH_MAX_INSTANCES	1618	X'00000652'
MQIACH_MAX_INSTS_PER_CLIENT	1619	X'00000653'
MQIACH_CLIENT_CHANNEL_WEIGHT	1620	X'00000654'
MQIACH_CONNECTION_AFFINITY	1621	X'00000655'
MQIACH_AUTH_INFO_TYPES	1622	X'00000656'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'
MQIACH_MSGS_RCVD	1634	X'00000662'
MQIACH_PENDING_OUT	1635	X'00000663'
MQIACH_AVAILABLE_CIPHERSPECS	1636	X'00000664'
MQIACH_MATCH	1637	X'00000665'
MQIACH_USER_SOURCE	1638	X'00000666'
MQIACH_WARNING	1639	X'00000667'
MQIACH_DEF_RECONNECT	1640	X'00000668'
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 V 9.1.3 MQIACH_SPL_PROTECTION	1646	X'0000066E'
MQIACH_LAST_USED	1646	X'0000066E'

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

Tabla 188. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_PRIMERO	701	X'000002BD'
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'

Tabla 188. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_BACKOUTS	704	X'000002C0'
MQIAMO_BROWSES	705	X'000002C1'
MQIAMO_BROWSE_MAX_BYTES	706	X'000002C2'
MQIAMO_BROWSE_MIN_BYTES	707	X'000002C3'
MQIAMO_BROWSES_FAILED	708	X'000002C4'
MQIAMO_CLOSES	709	X'000002C5'
MQIAMO_COMMITS	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
MQIAMO_CONNS	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
MQIAMO_DISCS	714	X'000002CA'
MQIAMO_DISCS_IMPLICIT	715	X'000002CB'
MQIAMO_DISC_TYPE	716	X'000002CC'
MQIAMO_EXIT_TIME_AVG	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
MQIAMO_EXIT_TIME_MIN	719	X'000002CF'
MQIAMO_FULL_BATCHES	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
MQIAMO_GETS	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
MQIAMO_GETS_FAILED	725	X'000002D5'
MQIAMO_INCOMPLETE_BATCHES	726	X'000002D6'
MQIAMO_INQS	727	X'000002D7'
MQIAMO_MSGS	728	X'000002D8'
MQIAMO_NET_TIME_AVG	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
MQIAMO_OBJECT_COUNT	732	X'000002DC'
MQIAMO_OPENS	733	X'000002DD'
MQIAMO_PUT1S	734	X'000002DE'
MQIAMO_PUTS	735	X'000002DF'
MQIAMO_PUT_MAX_BYTES	736	X'000002E0'
MQIAMO_PUT_MIN_BYTES	737	X'000002E1'
MQIAMO_PUT_RETRIES	738	X'000002E2'
MQIAMO_Q_MAX_DEPTH	739	X'000002E3'
MQIAMO_Q_MIN_DEPTH	740	X'000002E4'
MQIAMO_Q_TIME_AVG	741	X'000002E5'
MQIAMO_Q_TIME_MAX	742	X'000002E6'

Tabla 188. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_Q_TIME_MIN	743	X'000002E7'
MQIAMO_SETS	744	X'000002E8'
MQIAMO_CONNS_FAILED	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_FAILED	752	X'000002F0'
MQIAMO_SETS_FAILED	753	X'000002F1'
MQIAMO_PUTS_FAILED	754	X'000002F2'
MQIAMO_PUT1S_FAILED	755	X'000002F3'
MQIAMO_CLOSES_FAILED	757	X'000002F5'
MQIAMO_MSGS_EXPIRED	758	X'000002F6'
MQIAMO_MSGS_NOT_QUEUED	759	X'000002F7'
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
MQIAMO_SUBS_FAILED	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
MQIAMO_SUBRQS_FAILED	768	X'00000300'
MQIAMO_CBS	769	X'00000301'
MQIAMO_CBS_FAILED	770	X'00000302'
MQIAMO_CTLS	771	X'00000303'
MQIAMO_CTLS_FAILED	772	X'00000304'
MQIAMO_STATS	773	X'00000305'
MQIAMO_STATS_FAILED	774	X'00000306'
MQIAMO_SUB_DUR_HIGHWATER	775	X'00000307'
MQIAMO_SUB_DUR_LOWWATER	776	X'00000308'
MQIAMO_SUB_NDUR_HIGHWATER	777	X'00000309'
MQIAMO_SUB_NDUR_LOWWATER	778	X'0000030A'
MQIAMO_TOPIC_PUTS	779	X'0000030B'
MQIAMO_TOPIC_PUTS_FAILED	780	X'0000030C'
MQIAMO_TOPIC_PUT1S	781	X'0000030D'
MQIAMO_TOPIC_PUT1S_FAILED	782	X'0000030E'
MQIAMO_PUBLISH_MSG_COUNT	784	X'00000310'
MQIAMO_UNSUBS_DUR	786	X'00000312'
MQIAMO_UNSUBS_NDUR	787	X'00000313'
MQIAMO_UNSUBS_FAILED	788	X'00000314'
MQIAMO_INTERVALO	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'

Tabla 188. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_FEEDBACK_MODE	793	X'00000319'
MQIAMO_RELIABILITY_TYPE	794	X'0000031A'
MQIAMO_LATE_JOIN_MARK	795	X'0000031B'
MQIAMO_NACKS_RCVD	796	X'0000031C'
MQIAMO_REPAIR_PKTS	797	X'0000031D'
MQIAMO_HISTORY_PKTS	798	X'0000031E'
MQIAMO_PENDING_PKTS	799	X'0000031F'
MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE	801	X'00000321'
MQIAMO_MCAST_BATCH_TIME	802	X'00000322'
MQIAMO_MCAST_HEARTBEAT	803	X'00000323'
MQIAMO_DEST_DATA_PORT	804	X'00000324'
MQIAMO_DEST_REPAIR_PORT	805	X'00000325'
MQIAMO_ACKS_RCVD	806	X'00000326'
MQIAMO_ACTIVE_ACKERS	807	X'00000327'
MQIAMO_PKTS_SENT	808	X'00000328'
MQIAMO_TOTAL_REPAIR_PKTS	809	X'00000329'
MQIAMO_TOTAL_PKTS_SENT	810	X'0000032A'
MQIAMO_TOTAL_MSGS_SENT	811	X'0000032B'
MQIAMO_TOTAL_BYTES_SENT	812	X'0000032C'
MQIAMO_NUM_STREAMS	813	X'0000032D'
Comentarios de MQIAMO_ACK_FEEDBACK	814	X'0000032E'
MQIAMO_NACK_FEEDBACK	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'
MQIAMO_MSGS_RCVD	817	X'00000331'
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MSGS_ENTREGADO	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_CAÍDO	822	X'00000336'
MQIAMO_PKTS_DUPLICADAS	823	X'00000337'
MQIAMO_NACKS_CREATED	824	X'00000338'
MQIAMO_NACK_PKTS_SENT	825	X'00000339'
MQIAMO_REPAIR_PKTS_RQSTD	826	X'0000033A'
MQIAMO_REPAIR_PKTS_RCVD	827	X'0000033B'
MQIAMO_PKTS_REPAIRED	828	X'0000033C'
MQIAMO_TOTAL_MSGS_RCVD	829	X'0000033D'
MQIAMO_TOTAL_MSGS_BYTES_RCVD	830	X'0000033E'
MQIAMO_TOTAL_REPAIR_PKTS_RCVD	831	X'0000033F'

Tabla 188. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_TOTAL_REPAIR_PKTS_RQSTD	832	X'00000340'
MQIAMO_TOTAL_MSGS_PROCESSED	833	X'00000341'
MQIAMO_TOTAL_MSGS_SELECTED	834	X'00000342'
MQIAMO_TOTAL_MSGS_CADUCADO	835	X'00000343'
MQIAMO_TOTAL_MSGS_ENTREGADO	836	X'00000344'
MQIAMO_TOTAL_MSGS_DEVUELTO	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

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

Tabla 189. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO64_AVG_Q_TIME	703	X'000002BF'
MQIAMO64_Q_TIME_AVG	741	X'000002E5'
MQIAMO64_Q_TIME_MAX	742	X'000002E6'
MQIAMO64_Q_TIME_MIN	743	X'000002E7'
MQIAMO64_BROWSE_BYTES	745	X'000002E9'
MQIAMO64_BYTES	746	X'000002EA'
MQIAMO64_GET_BYTES	747	X'000002EB'
MQIAMO64_PUT_BYTES	748	X'000002EC'
MQIAMO64_TOPIC_PUT_BYTES	783	X'0000030F'
MQIAMO64_PUBLISH_MSG_BYTES	785	X'00000311'

MQIASY_ * (Selectores enteros del sistema)

Tabla 190. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIASY_FIRST	-1	X'FFFFFFFF'
MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
MQIASY_TYPE	-2	X'FFFFFFFE'
MQIASY_COMMAND	-3	X'FFFFFFFD'
MQIASY_MSG_SEQ_NUMBER	-4	X'FFFFFFFC'
CONTROL DE MQIASY_R	-5	X'FFFFFFFB'
CÓDIGO_EMPRESA	-6	X'FFFFFFFA'
MQIASY_REASON	-7	X'FFFFFFF9'
MQIASY_BAG_OPTIONS	-8	X'FFFFFFF8'
MQIASY_VERSION	-9	X'FFFFFFF7'
MQIASY_LAST_USED	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_ * (Autenticador de cabecera de información de IMS)

Tabla 191. Nombres y valores de constantes	
Nombre	**Valor**
MQIAUT_NONE	"¬¬¬¬¬¬¬¬"
MQIAUT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'

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

MQIAV_ * (Valores de atributo entero)

Tabla 192. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_UNDEFINED	-2	X'FFFFFFFE'

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

Tabla 193. Nombres y valores de constantes	
Nombre	**Valor**
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_ * (Formato de mandato, Opciones dudosas)

Tabla 194. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIDO_COMMIT	1	X'00000001'
MQIDO_BACKOUT	2	X'00000002'

MQIEP_ * (Puntos de entrada de interfaz)

Estructura de parámetros de seguridad de conexión

Tabla 195. Estructuras de constantes	
Nombre	**Estructura**
MQIEP_STRUC_ID	"IEP¬"
MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

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

Tabla 196. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_ * (Colas dentro del grupo)

Tabla 197. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIGQ_INHABILITADO	0	X'00000000'
MQIGQ_HABILITADO	1	X'00000001'

MQIGQPA_ * (Autoridad de colocación en cola dentro del grupo)

Tabla 198. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIGQPA_PREDETERMINADO	1	X'00000001'
CONTEXTO_MQIGQPA	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

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

Estructura de cabecera de información de IMS

Tabla 199. Estructuras de constantes	
Nombre	**Estructura**
MQIIH_STRUC_ID	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

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

Tabla 200. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

Distintivos de cabecera de información de IMS

Tabla 201. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIIH_NONE	0	X'00000000'
MQIIH_PASS_EXPIRATION	1	X'00000001'
MQIIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQIIH_REPLY_FORMAT_NONE	8	X'00000008'
MQIIH_IGNORE_PURG	16	X'00000010'
MQIIH_CM0_REQUEST_RESPONSE	32	X'00000020'

MQIMPO_ * (Consultar opciones y estructura de propiedad de mensaje)

Consultar estructura de opciones de propiedad de mensaje

Tabla 202. Estructuras de constantes

Nombre	Estructura
MQIM_ID_ESTRUCTURA	"IMPO"
MQIM_STRUC_ID_ARRAY	'I','M','P','O'

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

Tabla 203. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIMPO_VERSION_1	1	X'00000001'
MQIM_VERSIÓN_ACTUAL	1	X'00000001'

Consultar opciones de propiedad de mensaje

Tabla 204. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIM_CONVERT_TYPE	2	X'00000002'
MQIM_QUERY_LENGTH	4	X'00000004'
MQIM_INQ_PRIMERO	0	X'00000000'
MQIM_INQ_NEXT	8	X'00000008'
MQIM_INQ_PROP_UNDER_CURSOR	16	X'00000010'
MQIM_CONVERT_VALUE	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_ * (Formato de mandato, Disposiciones de entrada)

Tabla 205. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQINBD_Q_MGR	0	X'00000000'
MQINBD_XX_ENCODE_CASE_ONE grupo	3	X'00000003'

MQIND_ * (Valores de índice especiales)

Tabla 206. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_TODOS	-2	X'FFFFFFFE'

MQIPADDR_ * (Versiones de dirección IP)

Tabla 207. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIPADDR_IPv4	0	X'00000000'
MQIPADDR_IPv6	1	X'00000001'

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

Tabla 208. Nombres y valores de constantes

Nombre	Valor
MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_ * (Tipos de índice)

Tabla 209. Valores de constantes

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

MQITEM_ * (Tipo de elemento para mqInquireItemInfo)

Tabla 210. Valores de constantes

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

MQITII_ * (Identificador de instancia de transacción de cabecera de información de IMS)

Tabla 211. Nombres y valores de constantes

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

MQITS_ * (Estados de transacción de cabecera de información de IMS)

Tabla 212. Nombres y valores de constantes

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

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

MQKAI_ * (IntervaloKeepAlive)

Tabla 213. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_ * (Administración maestra)

Tabla 214. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

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

Tabla 215. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMCAS_STOPPED	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_ * (Tipos de MCA)

Tabla 216. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMCAT_PROCESO	1	X'00000001'
HEBRA MQMCAT_THREAD	2	X'00000002'

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

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

Tabla 217. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
VERSIÓN_CARPETA_MQMCD_MQM	1	X'00000001'

Nombres de etiqueta de opciones de publicación/suscripción

Tabla 218. Nombres y valores de constantes	
Nombre	**Valor**
MQMCD_MSG_DOMAIN	"Msd"
MQMCD_MSG_SET	"Set"
MQMCD_MSG_TYPE	"Type"
MQMCD_MSG_FORMAT	"Fmt"

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

Tabla 219. Nombres y valores de constantes

Nombre	Valor
MQMCD_MSG_DOMAIN_B	"<Msd>"
MQMCD_MSG_DOMAIN_E	"</Msd>"
MQMCD_MSG_SET_B	"<Set>"
MQMCD_MSG_SET_E	"</Set>"
MQMCD_MSG_TYPE_B	"<Type>"
MQMCD_MSG_TYPE_E	"</Type>"
MQMCD_MSG_FORMAT_B	"<Fmt>"
MQMCD_MSG_FORMAT_E	"</Fmt>"

Valores de etiqueta de opciones de publicación/suscripción

Tabla 220. Nombres y valores de constantes

Nombre	Valor
MQMCD_DOMAIN_NONE	"none"
MQMCD_DOMAIN_NEON	"neon"
MQMCD_DOMAIN_MRM	"mrm"
MQMCD_DOMAIN_JMS_NONE	"jms_none"
MQMCD_DOMAIN_JMS_TEXT	"jms_text"
MQMCD_DOMAIN_JMS_OBJECT	"jms_object"
MQMCD_DOMAIN_JMS_MAP	"jms_map"
MQMCD_DOMAIN_JMS_STREAM	"jms_stream"
MQMCD_DOMAIN_JMS_BYTES	"jms_bytes"

MQMD_ * (Estructura del descriptor de mensaje)

Tabla 221. Estructuras de constantes

Nombre	Estructura
MQMD_STRUC_ID	"MD¬¬"
MQMD_STRUC_ID_ARRAY	'M','D','¬','¬'

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

Tabla 222. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_ * (Estructura de extensión de descriptor de mensaje)

Tabla 223. Estructuras de constantes	
Nombre	**Estructura**
MQMDE_STRUC_ID	"MDE¬"
MQMDE_STRUC_ID_ARRAY	'M','D','E','¬'

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

Tabla 224. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDE_VERSION_2	2	X'00000002'
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_ * (Distintivos de extensión de descriptor de mensaje)

Tabla 225. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDEF_NONE	0	X'00000000'

MQMDS_ * (secuencia de entrega de mensajes)

Tabla 226. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDS_PRIORITY	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_ * (distintivos de mensaje)

Tabla 227. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMF_SEGMENTATION_INHIBIDO	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
SEGMENTO MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

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

Estructura de manejadores de mensajes para opciones de almacenamiento intermedio

Tabla 228. Estructuras de constantes

Nombre	Estructura
MQMHBO_STRUC_ID	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

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

Tabla 229. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

Opciones de manejador de mensajes a almacenamiento intermedio

Tabla 230. Valores de constantes

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

MQMI_ * (Identificador de mensaje)

Tabla 231. Nombres y valores de constantes

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

MQMMBI_ * (Marca de mensaje-Intervalo de examen)

Tabla 232. Valores de constantes

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

MQMO_ * (Opciones de coincidencia)

Tabla 233. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'

Tabla 233. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_ * (Formato de mandato, Opciones de modalidad)

Tabla 234. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMODE_FORCE	0	X'00000000'
MQMODE_QUIESCE	1	X'00000001'
MQMODE_TERMINAR	2	X'00000002'

MQMON_ * (Valores de supervisión)

Tabla 235. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMON_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQMON_NONE	-1	X'FFFFFFFF'
MQMON_Q_MGR	-3	X'FFFFFFFD'
MQMON_OFF	0	X'00000000'
MQMON_ON	1	X'00000001'
MQMON_INHABILITADO	0	X'00000000'
MQMON_HABILITADO	1	X'00000001'
MQMON_LOW	17	X'00000011'
MQMON_MEDIO	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_ * (Tipos de mensaje)

Tabla 236. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
MQMT_REPORT	4	X'00000004'
MQMT_MQE_FIELDS_FROM_MQE	112	X'00000070'
MQMT_MQE_FIELDS	113	X'00000071'
MQMT_SYSTEM_LAST	65535	X'0000FFFF'
MQMT_APPL_PRIMERO	65536	X'00010000'
MQMT_APPL_LAST	999999999	X'3B9AC9FF'

MQMTOK_ * (Señal de mensaje)

Tabla 237. Nombres y valores de constantes	
Nombre	**Valor**
MQMTOK_NONE	X'00...00' (16 nulos)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 nulos)

Tabla 238. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMTOK_NONE	X'00...00'	(16 nulls)
MQMTOK_NONE_ARRAY	'\0','\0',...	(16 nulls)

MQNC_ * (Recuento de nombres)

Tabla 239. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_ * (Clase de mensaje no persistente)

Tabla 240. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_ * (NonPersistent-XX_ENCODE_CASE_One Velocidades de mensajes)

Tabla 241. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_ * (Tipos de lista de nombres)

Tabla 242. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNT_NONE	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
MQNT_AUTH_INFO	4	X'00000004'
MQNT_ALL	1001	X'000003E9'

MQNVS_ * (Nombres para serie de nombre/valor)

Tabla 243. Nombres y valores de constantes	
Nombre	**Valor**
MQNVS_TIPO_APL	"OPT_APP_GRP¬"

Tabla 243. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQNVS_TIPO_MSGS	"OPT_MSG_TYPE¬"

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

MQOA_ * (Límites para selectores para atributos de objeto)

Tabla 244. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOA_PRIMERO	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_ * (estructura de descriptor de objeto)

Tabla 245. Estructuras de constantes	
Nombre	**Estructura**
MQOD_ID_STRUCD	"OD¬¬"
MQOD_STRUC_ID_ARRAY	'O','D','¬','¬'

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

Tabla 246. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
MQOD_CURRENT_VERSION	4	X'00000004'
MQOD_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

MQOII_ * (Identificador de instancia de objeto)

Tabla 247. Nombres y valores de constantes	
Nombre	**Valor**
MQOII_NONE	X'00...00' (24 nulos)
MQOII_NONE_ARRAY	'\0','\0',... (24 nulos)

MQOL_ * (Longitud original)

Tabla 248. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOL_UNDEFINED	-1	X'FFFFFFFF'

MQOM_ * (Opciones de mensajes de Db2 obsoletos en el grupo de consulta)

Tabla 249. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOM_NO	0	X'00000000'
MQOM_SÍ	1	X'00000001'

MQOO_ * (Opciones de apertura)

Tabla 250. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOO_BIND_AS_Q_DEF	0	X'00000000'
MQOO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQOO_INPUT_AS_Q_DEF	1	X'00000001'
MQOO_INPUT_SHARED	2	X'00000002'
MQOO_INPUT_EXCLUSIVE	4	X'00000004'
MQOO_BROWSE	8	X'00000008'
MQOO_OUTPUT	16	X'00000010'
MQOO_INQUIRE	32	X'00000020'
MQOO_SET	64	X'00000040'
MQOO_SAVE_ALL_CONTEXT	128	X'00000080'
MQOO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQOO_PASS_ALL_CONTEXT	512	X'00000200'
MQOO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQOO_SET_ALL_CONTEXT	2048	X'00000800'
MQOO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQOO_FAIL_IF_QUIESCING	8192	X'00002000'
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NOT_FIXED	32768	X'00008000'
MQOO_CO_OP	131072	X'00020000'
MQOO_RESOLVE_LOCAL_TOPIC	262144	X'00040000'
MQOO_NO_READ_AHEAD	524288	X'00080000'
MQOO_READ_AHEAD	1048576	X'00100000'
MQOO_BIND_ON_GROUP	4194304	X'00400000'

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

Tabla 251. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_ * (Códigos de operación para MQCTL y MQCB)

Códigos de operación para MQCTL

Tabla 252. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOP_START	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

Códigos de operación para MQCB

Tabla 253. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOP_REGISTRO	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Códigos de operación para MQCTL y MQCB

Tabla 254. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOP_SUSPENDER	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_ * (Valores relacionados con la estructura MQOPEN_PRIV)

Tabla 255. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_ * (Operaciones de actividad)

Tabla 256. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_DESCONOCIDO	0	X'00000000'
MQOPER_EXAMINAR	1	X'00000001'
MQOPER_DISCARD	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'

Tabla 256. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DESCARDED_PUBLISH	12	X'0000000C'
MQOPER_SYSTEM_LAST	65535	X'0000FFFF'
MQOPER_APPL_PRIMERO	65536	X'00010000'
MQOPER_APPL_LAST	999999999	X'3B9AC9FF'

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

Tipos de objeto

Tabla 257. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOT_NONE	0	X'00000000'
MQOT_Q	1	X'00000001'
MQOT_NAMELIST	2	X'00000002'
MQOT_PROCESS	3	X'00000003'
Clase de almacenamiento MQOT_STORAGE_CLASS	4	X'00000004'
MQOT_Q_MGR	5	X'00000005'
MQOT_CHANNEL	6	X'00000006'
MQOT_AUTH_INFO	7	X'00000007'
MQOT_TOPIC	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
MQOT_ESCUCHA	11	X'0000000B'
SERVICIO MQOT_SERVICE	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Tipos de objetos ampliados

Tabla 258. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOT_ALL	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODEL_Q	1003	X'000003EB'
MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTE_Q	1005	X'000003ED'
MQOT_SENDER_CHANNEL	1007	X'000003EF'
MQOT_SERVER_CHANNEL	1008	X'000003F0'
MQOT_CANAL_SOLICITANTE	1009	X'000003F1'
MQOT_RECEIVER_CHANNEL	1010	X'000003F2'
MQOT_CANAL_ACTUAL	1011	X'000003F3'
MQOT_SAVED_CHANNEL	1012	X'000003F4'

Tabla 258. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOT_SVRCONN_CHANNEL	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL	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_ * (Autorización de colocación)

Tabla 259. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPA_PREDETERMINADO	1	X'00000001'
CONTEXTO_MQPA	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_ * (Descriptor de propiedades, soporte y contexto)

Estructura del descriptor de propiedades

Tabla 260. Estructuras de constantes	
Nombre	**Estructura**
MQPD_STRUC_ID	"PD¬¬"
MQPD_STRUC_ID_ARRAY	'P','D','¬','¬'

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

Tabla 261. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

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

Opciones de descriptor de propiedad

Tabla 262. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPD_NONE	0	X'00000000'

Opciones de soporte de propiedad

Tabla 263. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPD_SUPPORT_OPTIONAL	1	X'00000001'
MQPD_SUPPORT_REQUIRED	1048576	X'00100000'
MQPD_SUPPORT_REQUIRED_IF_LOCAL	1024	X'00000400'
MQPD_REJECT_UNSUP_MASK	-1048576	X'FFF00000'
MQPD_ACCEPT_UNSUP_IF_XMIT_MASK	1047552	X'000FFC00'
MQPD_ACCEPT_UNSUP_MASK	1023	X'000003FF'

Contexto de propiedad

Tabla 264. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPD_NO_CONTEXT	0	X'00000000'
CONTEXTO_USUARIO_MQPDD	1	X'00000001'

MQPER_ * (Valores de persistencia)

Tabla 265. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPER_PERSISTENCE_AS_PARENT	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
MQPER_PERSISTENT	1	X'00000001'
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCE_AS_TOPIC_DEF	2	X'00000002'

MQPL_ * (Plataformas)

Tabla 266. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
MQPL_ZOS	1	X'00000001'
MQPL_OS2	2	X'00000002'
MQPL_AIX	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
MQPL_WINDOWS	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'
MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
MQPL_VM	18	X'00000012'

Tabla 266. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
DISPOSITIVO_MQPL_APPLIANCE	28	X'0000001C'
MQPL_NATIVE	1	X'00000001'

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

Estructura de opciones de colocación de mensaje

Tabla 267. Estructuras de constantes	
Nombre	**Estructura**
MQPMO_STRUC_ID	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

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

Tabla 268. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_CURRENT_VERSION	3	X'00000003'
MQPMO_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

Opciones de colocación de mensaje

Tabla 269. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
MQPMO_NEW_MSG_ID	64	X'00000040'
MQPMO_NEW_CORREL_ID	128	X'00000080'
MQPMO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQPMO_PASS_ALL_CONTEXT	512	X'00000200'
MQPMO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQPMO_SET_ALL_CONTEXT	2048	X'00000800'
MQPMO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQPMO_FAIL_IF_QUIESCING	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
MQPMO_LOGICAL_ORDER	32768	X'00008000'

Tabla 269. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_ASYNC_RESPONSE	65536	X'00010000'
MQPMO_SYNC_RESPONSE	131072	X'00020000'
MQPMO_RESOLVE_LOCAL_Q	262144	X'00040000'
MQPMO_RETAIN	2097152	X'00200000'
MQPMO_MD_FOR_OUTPUT_ONLY	8388608	X'00800000'
MQPMO_SCOPE_QMGR	67108864	X'04000000'
MQPMO_SUPPRESS_REPLYTO	134217728	X'08000000'
MQPMO_NOT_OWN_SUBS	268435456	X'10000000'
MQPMO_RESPONSE_AS_Q_DEF	0	X'00000000'
MQPMO_RESPONSE_AS_TOPIC_DEF	0	X'00000000'
MQPMO_NONE	0	X'00000000'

Opciones de colocación de mensaje para máscara de publicación

Tabla 270. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_ * (transferir campos de registro de mensajes)

Tabla 271. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMRF_MSG_ID	1	X'00000001'
ID MQPMRF_CORREL_ID	2	X'00000002'
MQPMRF_ID_grupo	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_ * (Formato de mandato, Opciones de depuración)

Tabla 272. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPO_SÍ	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_ * (Prioridad)

Tabla 273. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'

Tabla 273. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

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

Valores de control de propiedades de cola y canal

Tabla 274. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
COMPATIBILIDAD de MQPROP_COMPATIBILITY	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Longitud máxima propiedades

Tabla 275. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_ * (Colocar valores de respuesta)

Tabla 276. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_ * (Publicación/suscripción)

Formato de mandato Estado de publicación/suscripción

Tabla 277. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_DETENIENDO	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_RECHAZADO	6	X'00000006'

Etiquetas de publicación/suscripción como series

MQPS_COMMAND	"MQPSCommand"

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

Etiquetas de publicación/suscripción como series encerradas en blanco

MQPS_COMMAND_B	"¬MQPSCommand¬"
MQPS_COMP_CODE_B	"¬MQPSCompCode¬"
MQPS_ID_CORREL_B	"¬MQPSCorrelId¬"
MQPS_DELETE_OPTIONS_B	"¬MQPSDelOpts¬"
ID_ERROR_MQPS_B	"¬MQPSErrorId¬"
MQPS_ERROR_POS_B	"¬MQPSErrorPos¬"
MQPS_INTEGER_DATA_B	"¬MQPSIntData¬"
MQPS_ID_PARÁMETRO_B	"¬MQPSParmId¬"
MQPS_PUBCIONTION_OPTIONS_B	"¬MQPSPubOpts¬"
MQPS_PUBLISH_TIMESTAMP_B	"¬MQPSPubTime¬"
MQPS_Q_MGR_NAME_B	"¬MQPSQMgrName¬"
MQPS_Q_NAME_B	"¬MQPSQName¬"

MQPS_MOTIVO_B	"¬MQPSReason¬"
MQPS_MOTIVO_TEXTO_B	"¬MQPSReasonText¬"
MQPS_REGISTRATION_OPTIONS_B	"¬MQPSRegOpts¬"
MQPS_NÚMERO_SECUENCIA_B	"¬MQPSSeqNum¬"
MQPS_NOMBRE_SECUENCIA_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¬"
MQPS_ID_USER_B	"¬MQPSUserId¬"

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

Valores de etiqueta de mandato de publicación/suscripción como series

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

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

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

MQPS_DELETE_PUBCIONTION_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¬"

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

Valores de etiqueta de opciones de publicación/suscripción como series

MQPS_ADD_NAME	"AddName"

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

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

MQPS_ADD_NAME_B	"¬AddName¬"
MQPS_ANÓNIMO_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_PUBCIONTION_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_PUBLICATIONS_ONLY_B	"¬NewPubsOnly¬"
MQPS_NO_ALTERACIÓN_B	"¬NoAlter¬"
MQPS_NO_REGISTRATION_B	"¬NoReg¬"
MQPS_NON_PERSISTENT_B	"¬NonPers¬"
MQPS_NONE_B	"¬None¬"
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"¬OtherSubsOnly¬"
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_PUBCIONTION_B	"¬RetainPub¬"
MQPS_VARIABLE_USER_ID_B	"¬VariableUserId¬"

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

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

Tabla 278. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPSC_FOLDER_VERSION	1	X'00000001'

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

MQPSC_COMMAND	"Command"
MQPSC_REGISTRATION_OPTION	"RegOpt"
MQPSC_PUBCIONTION_OPTION	"PubOpt"
MQPSC_DELETE_OPTION	"DelOpt"
MQPSC_TOPIC	"Topic"
MQPSC_SUBSCRIPTION_POINT	"SubPoint"
FILTRO MQPSC	"Filter"
MQPSC_Q_MGR_NAME	"QMgrName"
MQPSC_Q_NAME	"QName"

MQPSC_PUBLISH_TIMESTAMP	"PubTime"
NÚMERO_SECUENCIA_MQPSC_NUMBER	"SeqNum"
MQPSC_SUBSCRIPTION_NAME	"SubName"
MQPSC_SUSCRIPTION_IDENTITY	"SubIdentity"
MQPSC_SUBSCRIPTION_USER_DATA	"SubUserData"
MQPSC_CORREL_ID	"CorrelId"

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

MQPSC_COMMAND_B	"<Command>"
MQPSC_COMMAND_E	"</Command>"
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"
MQPSC_PUBCIONTION_OPTION_B	"<PubOpt>"
MQPSC_PUBCIONTION_OPTION_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E	"</DelOpt>"
MQPSC_TOPIC_B	"<Topic>"
MQPSC_TOPIC_E	"</Topic>"
MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"
FILTER_MQPSC_B	"<Filter>"
FILTER_MQPS_E	"</Filter>"
MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NAME_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
MQPSC_NÚMERO_SECUENCIA_B	"<SeqNum>"
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"
MQPSC_SUBSCRIPTION_NAME_B	"<SubName>"
MQPSC_SUBSCRIPTION_NAME_E	"</SubName>"
MQPSC_SUBSCRIPTION_IDENTITY_B	"<SubIdentity>"
MQPSC_SUBSCRIPTION_IDENTITY_E	"</SubIdentity>"
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"
MQPSC_SUBSCRIPTION_USER_DATA_E	"</SubUserData>"
MQPSC_CORREL_ID_B	"<CorrelId>"

MQPSC_CORREL_ID_E	"</CorrelId>"

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

MQPSC_DELETE_PUBCIONCIÓN	"DeletePub"
MQPSC_DEREGISTER_SUBSCRIBER	"DeregSub"
MQPSC_PUBLICAR	"Publish"
MQPSC_REGISTER_SUSCRIPTOR	"RegSub"
MQPSC_REQUEST_UPDATE	"ReqUpdate"

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

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

MQPSCR_ * (Opciones de publicación/suscripción)

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

Tabla 279. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
VERSIÓN_CARPETA_MQPSCRP	1	X'00000001'

Nombres de etiqueta de opciones de publicación/suscripción

COMPLETACIÓN_MQPSCR_COMPLETAR	"Completion"
MQPSCR_RESPONSE	"Response"
RAZÓN_MQPSCR_RAZÓN	"Reason"

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

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

Valores de etiqueta de opciones de publicación/suscripción

MQPSCR_Correcto	"ok"
MQPSCR_AVISO	"warning"
ERROR DE MQPSRC	"error"

MQPSM_ * (modalidad Pub/Sub)

Tabla 280. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPSM_INHABILITADO	0	X'00000000'
MQPSM_COMPAT	1	X'00000001'
MQPSM_HABILITADO	2	X'00000002'

MQPSPROP_ * (Propiedades de mensaje de publicación/suscripción)

Tabla 281. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPAT	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

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

Tabla 282. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPSST_ALL	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PARENT	2	X'00000002'
MQPSST_HIJO	3	X'00000003'

MQPUBO_ * (Opciones de publicación/suscripción)

Tabla 283. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLICACIÓN	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRO	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLICATION	16	X'00000010'

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

Tabla 284. Estructuras de constantes

Nombre	Estructura
MQPXP_STRUC_ID	"PXP¬"
MQPXP_STRUC_ID_ARRAY	'P','X','P','¬'

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

Tabla 285. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPXP_VERSION_1	1	X'00000001'
MQPXP_CURRENT_VERSION	1	X'00000001'

MQQA_ * (Atributos de cola)

Inhibir obtención de valores

Tabla 286. Valores de constantes

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

Inhibir valores de colocación

Tabla 287. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQA_PUT_INHIBITED	1	X'00000001'
MQQA_PUT_ALLOWED	0	X'00000000'

Compartición de cola

Tabla 288. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Refuerzo de retrotracción

Tabla 289. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_BACKOUT_NOT_HARTIZADO	0	X'00000000'

MQQDT_ * (Tipos de definición de cola)

Tabla 290. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_ * (distintivos de cola)

Tabla 291. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

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

Tabla 292. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQMDT_EXPLICIT_CLUSTER_SENDER	1	X'00000001'
MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
MQQMDT_CLUSTER_RECEIVER	3	X'00000003'

MQQMF_ * (distintivos de gestor de colas)

Tabla 293. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQMF_REPOSITORY_Q_MGR	2	X'00000002'
MQQMF_CLUSSDR_USER_DEFINED	8	X'00000008'
MQQMF_CLUSSDR_AUTO_DEFINED	16	X'00000010'
MQQMF_AVAILABLE	32	X'00000020'

MQQMFAC_ * (Formato de mandato, Recurso de gestor de colas)

Tabla 294. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQMFAC_IMS_BRIDGE	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

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

Tabla 295. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQMSTA_INICIANDO	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA_QUIESCING	3	X'00000003'

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

Tabla 296. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQMT_NORMAL	0	X'00000000'
REPOSITORIO_MQQM	1	X'00000001'

MQQO_ * (Formato de mandato, Opciones de inmovilización)

Tabla 297. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQO_SÍ	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQSGD_ * (disposiciones de grupo de compartición de colas)

Tabla 298. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
MQQSGD_XX_ENCODE_CASE_ONE grupo	3	X'00000003'

Tabla 298. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSGD_PRIVADO	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_ * (Estado de grupo de compartición de colas con formato de mandato)

Tabla 299. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSGS_DESCONOCIDO	0	X'00000000'
MQQSGS_CREADO	1	X'00000001'
MQQSGS_ACTIVE	2	X'00000002'
MQQSGS_INACTIVE	3	X'00000003'
MQQSGS_FAILED	4	X'00000004'
MQQSGS_PENDIENTE	5	X'00000005'

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

Tabla 300. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

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

Tabla 301. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSO_NO	0	X'00000000'
MQQSO_SÍ	1	X'00000001'
MQQSO_COMPARTIDO	1	X'00000001'
MQQSO_EXCLUSIVE	2	X'00000002'

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

Tabla 302. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSOT_ALL	1	X'00000001'
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

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

Tabla 303. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSUM_SÍ	1	X'00000001'
MQQSUM_NO	0	X'00000000'

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

Tipos de cola

Tabla 304. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQT_LOCAL	1	X'00000001'
MQQT_MODEL	2	X'00000002'
MQQT_ALIAS	3	X'00000003'
MQQT_REMOTE	6	X'00000006'
MQQT_CLUSTER	7	X'00000007'

Tipos de cola ampliada

Tabla 305. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQT_ALL	1001	X'000003E9'

MQRC_ * (códigos de razón)

Tabla 306. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRC_NONE	0	X'00000000'
MQRC_APPL_PRIMERO	900	X'00000384'
MQRC_APPL_LAST	999	X'000003E7'
MQRC_ALIAS_BASE_Q_TYPE_ERROR	2001	X'000007D1'
MQRC_ALREADY_CONNECTED	2002	X'000007D2'
MQRC_BACKED_OUT	2003	X'000007D3'
MQRC_BUFFER_ERROR	2004	X'000007D4'
MQRC_BUFFER_LENGTH_ERROR	2005	X'000007D5'
MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
MQRC_CHAR_ATTRS_ERROR	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
MQRC_CONNECTION_BROKEN	2009	X'000007D9'
MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
ERROR_ENTORNO_MQRC	2012	X'000007DC'
MQRC_EXPIRY_ERROR	2013	X'000007DD'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
MQRC_HCONN_ERROR	2018	X'000007E2'
MQRC_HOBJ_ERROR	2019	X'000007E3'
MQRC_INHIBIDOR_VALOR_ERROR	2020	X'000007E4'
MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_ALCANZADO	2025	X'000007E9'
MQRC_MD_ERROR	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q	2027	X'000007EB'
MQRC_MSG_TYPE_ERROR	2029	X'000007ED'
MQRC_MSG_TOO_BIG_FOR_Q	2030	X'000007EE'
MQRC_MSG_TOO_BIG_FOR_Q_MGR	2031	X'000007EF'
MQRC_NO_MSG_AVAILABLE	2033	X'000007F1'
MQRC_NO_MSG_UNDER_CURSOR	2034	X'000007F2'
MQRC_NOT_AUTHORIZED	2035	X'000007F3'
MQRC_NOT_OPEN_FOR_BROWSE	2036	X'000007F4'
MQRC_NOT_OPEN_FOR_INPUT	2037	X'000007F5'
MQRC_NOT_OPEN_FOR_INQUIRE	2038	X'000007F6'
MQRC_NOT_OPEN_FOR_OUTPUT	2039	X'000007F7'
MQRC_NOT_OPEN_FOR_SET	2040	X'000007F8'
MQRC_OBJECT_CHANGED	2041	X'000007F9'
MQRC_OBJECT_IN_USE	2042	X'000007FA'
MQRC_OBJECT_TYPE_ERROR	2043	X'000007FB'
MQRC_OD_ERROR	2044	X'000007FC'
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
MQRC_OPTIONS_ERROR	2046	X'000007FE'
MQRC_PERSISTENCE_ERROR	2047	X'000007FF'
MQRC_PERSISTENT_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_EXCEEDS_MAXIMUM	2049	X'00000801'
MQRC_PRIORITY_ERROR	2050	X'00000802'
MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2051	X'00000803'
MQRC_Q_DELETED	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_Q_TYPE_ERROR	2057	X'00000809'
MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
MQRC_REPORT_OPTIONS_ERROR	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
MQRC_SECURITY_ERROR	2063	X'0000080F'
MQRC_SELECTOR_COUNT_ERROR	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TYPE	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NOT_AVAILABLE	2071	X'00000817'
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
MQRC_TRIGGER_CONTROL_ERROR	2075	X'0000081B'
MQRC_TRIGGER_DEPTH_ERROR	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
MQRC_TRIGGER_TYPE_ERROR	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
MQRC_TRUNCATED_MSG_FAILED	2080	X'00000820'
MQRC_UNKNOWN_ALIAS_BASE_Q	2082	X'00000822'
MQRC_UNKNOWN_OBJECT_NAME	2085	X'00000825'
MQRC_UNKNOWN_OBJECT_Q_MGR	2086	X'00000826'
MQRC_UNKNOWN_REMOTE_Q_MGR	2087	X'00000827'
MQRC_WAIT_INTERVAL_ERROR	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERROR	2092	X'0000082C'
MQRC_NOT_OPEN_FOR_PASS_ALL	2093	X'0000082D'
MQRC_NOT_OPEN_FOR_PASS_IDENT	2094	X'0000082E'
MQRC_NOT_OPEN_FOR_SET_ALL	2095	X'0000082F'
MQRC_NOT_OPEN_FOR_SET_IDENT	2096	X'00000830'
MQRC_CONTEXT_HANDLE_ERROR	2097	X'00000831'
MQRC_CONTEXT_NOT_AVAILABLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJECT_ALREADY_EXISTS	2100	X'00000834'
MQRC_OBJECT_DAMAGED	2101	X'00000835'
MQRC_RESOURCE_PROBLEM	2102	X'00000836'
MQRC_ANOTHER_Q_MGR_CONNECTED	2103	X'00000837'
MQRC_UNKNOWN_REPORT_OPTION	2104	X'00000838'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_STORAGE_CLASS_ERROR	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELADO	2107	X'0000083B'
MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
MQRC_FORMAT_ERROR	2110	X'0000083E'
MQRC_SOURCE_CCSID_ERROR	2111	X'0000083F'
MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR	2114	X'00000842'
MQRC_TARGET_CCSID_ERROR	2115	X'00000843'
MQRC_TARGET_INTEGER_ENC_ERROR	2116	X'00000844'
MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'
MQRC_TARGET_FLOAT_ENC_ERROR	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_TRUNCADO	2120	X'00000848'
MQRC_NO_EXTERNAL_PARTICIPANTES	2121	X'00000849'
MQRC_PARTICIPANT_NOT_AVAILABLE	2122	X'0000084A'
MQRC_OUTCOME_MIXED	2123	X'0000084B'
MQRC_OUTCOME_PENDING	2124	X'0000084C'
MQRC_BRIDGE_STARTED	2125	X'0000084D'
MQRC_BRIDGE_STOPPED	2126	X'0000084E'
MQRC_ADAPTER_STORAGE_INSUFICIENTE	2127	X'0000084F'
MQRC_UOW_IN_PROGRESS	2128	X'00000850'
MQRC_ADAPTER_CONN_LOAD_ERROR	2129	X'00000851'
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
nMQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_ERROR	2134	X'00000856'
MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_RAZONES	2136	X'00000858'
MQRC_OPEN_FAILED	2137	X'00000859'
MQRC_ADAPTER_DISC_LOAD_ERROR	2138	X'0000085A'
MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED	2140	X'0000085C'
MQRC_DLH_ERROR	2141	X'0000085D'
MQRC_HEADER_ERROR	2142	X'0000085E'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
MQRC_TARGET_LENGTH_ERROR	2144	X'00000860'
MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
MQRC_IIH_ERROR	2148	X'00000864'
MQRC_PCF_ERROR	2149	X'00000865'
MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
MQRC_OBJECT_RECORDS_ERROR	2155	X'0000086B'
MQRC_RESPONSE_RECORDS_ERROR	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
MQRC_PUT_MSG_RECORDS_ERROR	2159	X'0000086F'
MQRC_CONN_ID_IN_USE	2160	X'00000870'
MQRC_Q_MGR_QUIESCING	2161	X'00000871'
MQRC_Q_MGR_STOPPING	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_RESTRICCIÓN_PUENTE	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
ERROR MQRC_TMC	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE	2194	X'00000892'
MQRC_UNEXPECTED_ERROR	2195	X'00000893'
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'
MQRC_DEF_XMIT_Q_USAGE_ERROR	2199	X'00000897'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_MSG_MARKED_BROWSE_CO_OP	2200	X'00000898'
MQRC_NAME_IN_USE	2201	X'00000899'
MQRC_CONNECTION_QUIESCING	2202	X'0000089A'
MQRC_CONNECTION_STOPPING	2203	X'0000089B'
MQRC_ADAPTER_NOT_AVAILABLE	2204	X'0000089C'
MQRC_MSG_ID_ERROR	2206	X'0000089E'
MQRC_CORREL_ID_ERROR	2207	X'0000089F'
MQRC_FILE_SYSTEM_ERROR	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
MQRC_SOAP_DOTNET_ERROR	2210	X'000008A2'
MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
MQRC_SOAP_URL_ERROR	2212	X'000008A4'
MQRC_FILE_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHANNEL	2218	X'000008AA'
MQRC_CALL_IN_PROGRESS	2219	X'000008AB'
MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_ACTIVE	2222	X'000008AE'
MQRC_Q_MGR_NOT_ACTIVE	2223	X'000008AF'
MQRC_Q_DEPTH_HIGH	2224	X'000008B0'
MQRC_Q_DEPTH_LOW	2225	X'000008B1'
MQRC_Q_SERVICE_INTERVAL_HIGH	2226	X'000008B2'
MQRC_Q_SERVICE_INTERVAL_OK	2227	X'000008B3'
MQRC_RFH_HEADER_FIELD_ERROR	2228	X'000008B4'
MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIT_OF_WORK_NOT_STARTED	2232	X'000008B8'
MQRC_CHANNEL_AUTO_DEF_OK	2233	X'000008B9'
MQRC_CHANNEL_AUTO_DEF_ERROR	2234	X'000008BA'
MQRC_CFH_ERROR	2235	X'000008BB'
MQRC_CFIL_ERROR	2236	X'000008BC'
MQRC_CFIN_ERROR	2237	X'000008BD'
MQRC_CFSL_ERROR	2238	X'000008BE'
MQRC_CFST_ERROR	2239	X'000008BF'
MQRC_INCOMPLETE_GROUP	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'
MQRC_INCONSISTENT_CCSIDS	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
MQRC_INCONSISTENT_UOW	2245	X'000008C5'
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_MATCH_OPTIONS_ERROR	2247	X'000008C7'
MQRC_MDE_ERROR	2248	X'000008C8'
MQRC_MSG_FLAGS_ERROR	2249	X'000008C9'
MQRC_MSG_SEQ_NUMBER_ERROR	2250	X'000008CA'
ERROR DE MQRC_OFFSET_	2251	X'000008CB'
MQRC_ORIGINAL_LENGTH_ERROR	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NOT_AVAILABLE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE	2259	X'000008D3'
MQRC_XQH_ERROR	2260	X'000008D4'
MQRC_SRC_ENV_ERROR	2261	X'000008D5'
MQRC_SRC_NAME_ERROR	2262	X'000008D6'
MQRC_DEST_ENV_ERROR	2263	X'000008D7'
MQRC_DEST_NAME_ERROR	2264	X'000008D8'
ERROR MQRC_TM	2265	X'000008D9'
MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2268	X'000008DC'
MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_USE	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
MQRC_CONNECTION_ERROR	2273	X'000008E1'
MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
MQRC_CD_ERROR	2277	X'000008E5'
MQRC_CLIENT_CONN_ERROR	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
MQRC_HCONFIG_ERROR	2280	X'000008E8'
MQRC_FUNCTION_ERROR	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
MQRC_TERMINATION_FAILED	2287	X'000008EF'
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_SERVICE_ERROR	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
MQRC_USER_ID_NOT_AVAILABLE	2291	X'000008F3'
MQRC_UNKNOWN_ENTITY	2292	X'000008F4'
MQRC_UNKNOWN_AUTH_ENTITY	2293	X'000008F5'
MQRC_UNKNOWN_REF_OBJECT	2294	X'000008F6'
MQRC_CHANNEL_ACTIVATED	2295	X'000008F7'
MQRC_CHANNEL_NOT_ACTIVATED	2296	X'000008F8'
MQRC_UOW_CANCELADO	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED	2298	X'000008FA'
MQRC_SELECTOR_TYPE_ERROR	2299	X'000008FB'
MQRC_COMMAND_TYPE_ERROR	2300	X'000008FC'
MQRC_MULTIPLE_INSTANCE_ERROR	2301	X'000008FD'
MQRC_SYSTEM_ITEM_NOT_ALTERABLE	2302	X'000008FE'
MQRC_BAG_CONVERSION_ERROR	2303	X'000008FF'
MQRC_SELECTOR_OUT_OF_RANGE	2304	X'00000900'
MQRC_SELECTOR_NOT_UNIQUE	2305	X'00000901'
MQRC_INDEX_NOT_PRESENT	2306	X'00000902'
MQRC_STRING_ERROR	2307	X'00000903'
MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENT	2309	X'00000905'
MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_TRUNCATED	2311	X'00000907'
MQRC_SELECTOR_WRONG_TYPE	2312	X'00000908'
MQRC_INCONSISTENT_ITEM_TYPE	2313	X'00000909'
MQRC_INDEX_ERROR	2314	X'0000090A'
MQRC_SYSTEM_BAG_NOT_ALTERABLE	2315	X'0000090B'
MQRC_ITEM_COUNT_ERROR	2316	X'0000090C'
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPPORTED	2318	X'0000090E'
MQRC_ITEM_VALOR_ERROR	2319	X'0000090F'
MQRC_HBAG_ERROR	2320	X'00000910'
MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE	2322	X'00000912'
MQRC_STRING_LENGTH_ERROR	2323	X'00000913'
MQRC_INQUIRY_COMMAND_ERROR	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPPORTED	2325	X'00000915'
MQRC_BAG_TIPO_INCORRECTO	2326	X'00000916'
MQRC_ITEM_TYPE_ERROR	2327	X'00000917'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_SYSTEM_BAG_NOT_DELETABLE	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
MQRC_CODED_CHAR_SET_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
MQRC_WIH_ERROR	2333	X'0000091D'
MQRC_RFH_ERROR	2334	X'0000091E'
MQRC_RFH_STRING_ERROR	2335	X'0000091F'
MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
MQRC_RFH_PARM_MISSING	2339	X'00000923'
MQRC_CHAR_CONVERSION_ERROR	2340	X'00000924'
MQRC_UCS2_CONVERSION_ERROR	2341	X'00000925'
MQRC_DB2_NOT_AVAILABLE	2342	X'00000926'
MQRC_OBJECT_NOT_UNIQUE	2343	X'00000927'
MQRC_CONN_TAG_NOT_LIBERADO	2344	X'00000928'
MQRC_CF_NOT_AVAILABLE	2345	X'00000929'
MQRC_CF_STRUC_IN_USE	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_FAILED	2348	X'0000092C'
MQRC_CF_STRUC_ERROR	2349	X'0000092D'
MQRC_CONN_TAG_NOT_USABLE	2350	X'0000092E'
MQRC_GLOBAL_UOW_CONFLICT	2351	X'0000092F'
MQRC_LOCAL_UOW_CONFLICT	2352	X'00000930'
MQRC_HANDLE_IN_USE_FOR_UOW	2353	X'00000931'
MQRC_UOW_ENLISTMENT_ERROR	2354	X'00000932'
MQRC_UOW_MIX_NOT_SUPPORTED	2355	X'00000933'
MQRC_WXP_ERROR	2356	X'00000934'
MQRC_CURRENT_RECORD_ERROR	2357	X'00000935'
MQRC_NEXT_OFFSET_ERROR	2358	X'00000936'
MQRC_NO_RECORD_AVAILABLE	2359	X'00000937'
MQRC_OBJECT_LEVEL_INCOMPATIBLE	2360	X'00000938'
MQRC_NEXT_RECORD_ERROR	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_CONTACTADO	2362	X'0000093A'
MQRC_MSG_NOT_MATCHED	2363	X'0000093B'
MQRC_JMS_FORMAT_ERROR	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED	2365	X'0000093D'
MQRC_ERR_CF_LEVEL	2366	X'0000093E'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_CONFIG_CREATE_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_PARTICIPANT_NOT_DEFINED	2372	X'00000944'
MQRC_CF_STRUC_FAILED	2373	X'00000945'
MQRC_API_EXIT_ERROR	2374	X'00000946'
MQRC_API_EXIT_INIT_ERROR	2375	X'00000947'
MQRC_API_EXIT_TERM_ERROR	2376	X'00000948'
MQRC_EXIT_REASON_ERROR	2377	X'00000949'
MQRC_RESERVED_VALUE_ERROR	2378	X'0000094A'
MQRC_NO_DATA_AVAILABLE	2379	X'0000094B'
MQRC_SCO_ERROR	2380	X'0000094C'
MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
MQRC_CRYPTO_HARDWARE_ERROR	2382	X'0000094E'
MQRC_AUTH_INFO_REC_COUNT_ERROR	2383	X'0000094F'
MQRC_AUTH_INFO_REC_ERROR	2384	X'00000950'
MQRC_AIR_ERROR	2385	X'00000951'
MQRC_AUTH_INFO_TYPE_ERROR	2386	X'00000952'
MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
MQRC_LDAP_USER_NAME_ERROR	2388	X'00000954'
MQRC_LDAP_USER_NAME_LENGTH_ERR	2389	X'00000955'
MQRC_LDAP_PASSWORD_ERROR	2390	X'00000956'
MQRC_SSL_ALREADY_INITIALIZED	2391	X'00000957'
MQRC_SSL_CONFIG_ERROR	2392	X'00000958'
MQRC_SSL_INITIALIZATION_ERROR	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
MQRC_CFBS_ERROR	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED	2396	X'0000095C'
MQRC_JSSE_ERROR	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
MQRC_SSL_PEER_NAME_ERROR	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_REVOKED	2401	X'00000961'
MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'
MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'
MQRC_UOW_COMMITTED	2408	X'00000968'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_SSL_KEY_RESET_ERROR	2409	X'00000969'
MQRC_UNKNOWN_COMPONENT_NAME	2410	X'0000096A'
MQRC_LOGGER_STATUS	2411	X'0000096B'
MQRC_COMMAND_MQSC	2412	X'0000096C'
MQRC_COMMAND_PCF	2413	X'0000096D'
MQRC_CFIF_ERROR	2414	X'0000096E'
MQRC_CFSF_ERROR	2415	X'0000096F'
MQRC_CFGR_ERROR	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_GROUP	2417	X'00000971'
MQRC_FILTER_OPERATOR_ERROR	2418	X'00000972'
MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
MQRC_EPH_ERROR	2420	X'00000974'
MQRC_RFH_FORMAT_ERROR	2421	X'00000975'
MQRC_CFBF_ERROR	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICTO	2423	X'00000977'
MQRC_SD_ERROR	2424	X'00000978'
MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
MQRC_STS_ERROR	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE	2429	X'0000097D'
MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_ERROR	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
MQRC_ALTER_SUB_ERROR	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG	2437	X'00000985'
MQRC_SRO_ERROR	2438	X'00000986'
MQRC_SUB_NAME_ERROR	2440	X'00000988'
MQRC_OBJECT_STRING_ERROR	2441	X'00000989'
MQRC_PROPERTY_NAME_ERROR	2442	X'0000098A'
MQRC_SEGMENTATION_NOT_ALLOWED	2443	X'0000098B'
MQRC_CBD_ERROR	2444	X'0000098C'
MQRC_CTLO_ERROR	2445	X'0000098D'
MQRC_NO_CALLBACKS_ACTIVE	2446	X'0000098E'
MQRC_CALLBACK_NOT_REGISTERED	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'
MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_HMSG_ERROR	2460	X'0000099C'
MQRC_CMHO_ERROR	2461	X'0000099D'
MQRC_DMHO_ERROR	2462	X'0000099E'
MQRC_SMPO_ERROR	2463	X'0000099F'
MQRC_IMPO_ERROR	2464	X'000009A0'
MQRC_PROPERTY_NAME_TOO_BIG	2465	X'000009A1'
MQRC_PROP_VALUE_NOT_CONVERTED	2466	X'000009A2'
MQRC_PROP_TYPE_NOT_SUPPORTED	2467	X'000009A3'
MQRC_PROPERTY_VALUE_TOO_BIG	2469	X'000009A5'
MQRC_PROP_CONV_NOT_SUPPORTED	2470	X'000009A6'
MQRC_PROPERTY_NOT_AVAILABLE	2471	X'000009A7'
MQRC_PROP_NUMBER_FORMAT_ERROR	2472	X'000009A8'
ERROR_TIPO_PROPIEDAD_MQRC	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NO_RETENIDO	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CAMBIADO	2480	X'000009B0'
MQRC_DMPO_ERROR	2481	X'000009B1'
MQRC_PD_ERROR	2482	X'000009B2'
MQRC_CALLBACK_TYPE_ERROR	2483	X'000009B3'
MQRC_CBD_OPTIONS_ERROR	2484	X'000009B4'
MQRC_MAX_MSG_LENGTH_ERROR	2485	X'000009B5'
MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
MQRC_CALLBACK_LINK_ERROR	2487	X'000009B7'
MQRC_OPERATION_ERROR	2488	X'000009B8'
MQRC_BMHO_ERROR	2489	X'000009B9'
MQRC_UNSUPPORTED_PROPERTY	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERT	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'
MQRC_MODULE_NOT_FOUND	2495	X'000009BF'
MQRC_MODULE_INVALID	2496	X'000009C0'
MQRC_MODULE_ENTRY_NOT_FOUND	2497	X'000009C1'
MQRC_MIXED_CONTENT_NOT_ALLOWED	2498	X'000009C2'
MQRC_MSG_HANDLE_IN_USE	2499	X'000009C3'
MQRC_HCONN_ASYNC_ACTIVE	2500	X'000009C4'
MQRC_MHBO_ERROR	2501	X'000009C5'
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_XX_ENCODE_CASE_ONE inhibida	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'
MQRC_XEPO_ERROR	2507	X'000009CB'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQRC_GROUPING_NOT_ALTERABLE	2515	X'000009D3'
MQRC_SELECTOR_INVALID_FOR_TYPE	2516	X'000009D4'
MQRC_HOBJ_QUIESCED	2517	X'000009D5'
MQRC_HOBJ_QUIESCED_NO_MSGS	2518	X'000009D6'
MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
MQRC_RES_OBJECT_STRING_ERROR	2520	X'000009D8'
MQRC_CONNECTION_SUSPENDED	2521	X'000009D9'
MQRC_INVALID_DESTINATION	2522	X'000009DA'
MQRC_INVALID_SUBSCRIPTION	2523	X'000009DB'
MQRC_SELECTOR_NOT_ALTERABLE	2524	X'000009DC'
MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DELIVERED	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_STOPPED	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICTO	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICTO	2530	X'000009E2'
MQRC_PUBSUB_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERATION_NOT_ALLOWED	2534	X'000009E6'
MQRC_ACTION_ERROR	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NOT_AVAILABLE	2538	X'000009EA'
MQRC_CHANNEL_CONFIG_ERROR	2539	X'000009EB'
MQRC_UNKNOWN_CHANNEL_NAME	2540	X'000009EC'
MQRC_LOOPING_PUBLICATION	2541	X'000009ED'
MQRC_ALREADY_JOINED	2542	X'000009EE'
MQRC_CHANNEL_SSL_WARNING	2552	X'000009F8'
MQRC_OCSP_URL_ERROR	2553	X'000009F9'
MQRC_CIPHER_SPEC_NOT_SUITE_B	2591	X'00000A1F'
MQRC_SUITE_B_ERROR	2592	X'00000A20'
MQRC_PASSWORD_PROTECCION_ERROR	2594	X'00000A22'
MQRC_REOPEN_EXCL_INPUT_ERROR	6100	X'000017D4'
MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'

Tabla 306. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
MQRC_REOPEN_TEMPORARY_Q_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED	6104	X'000017D8'
MQRC_CURSOR_NOT_VALID	6105	X'000017D9'
MQRC_ENCODING_ERROR	6106	X'000017DA'
MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_NULL_POINTER	6108	X'000017DC'
MQRC_NO_CONNECTION_REFERENCE	6109	X'000017DD'
MQRC_NO_BUFFER	6110	X'000017DE'
MQRC_BINARY_DATA_LENGTH_ERROR	6111	X'000017DF'
MQRC_BUFFER_NOT_AUTOMATIC	6112	X'000017E0'
MQRC_INSUFICIENT_BUFFER	6113	X'000017E1'
MQRC_INSUFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_TRUNCADO	6115	X'000017E3'
MQRC_LONGITUD_CERO	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
DESPLAZAMIENTO_NEGATIVO_MQRC	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
MQRC_CONTEXT_OPEN_ERROR	6122	X'000017EA'
MQRC_STRUC_LENGTH_ERROR	6123	X'000017EB'
MQRC_NOT_CONNECTED	6124	X'000017EC'
MQRC_NOT_OPEN	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
MQRC_INCONSISTENT_OPEN_OPTIONS	6127	X'000017EF'
VERSIÓN_ERROR_MQRC	6128	X'000017F0'
ERROR_REFERENCIA_MQRC	6129	X'000017F1'

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

Consulte Códigos de razón PCF para obtener más información sobre la respuesta del programador.

Tabla 307. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CFH_TYPE_ERROR	3001	X'00000BB9'
MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
MQRCCF_CFH_VERSION_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_SEQ_NUMBER_ERR	3004	X'00000BBC'
MQRCCF_CFH_CONTROL_ERROR	3005	X'00000BBD'
MQRCCF_CFH_PARM_COUNT_ERROR	3006	X'00000BBE'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CFH_COMMAND_ERROR	3007	X'00000BBF'
MQRCCF_COMMAND_FAILED	3008	X'00000BC0'
MQRCCF_CFIN_LENGTH_ERROR	3009	X'00000BC1'
MQRCCF_CFST_LENGTH_ERROR	3010	X'00000BC2'
MQRCCF_CFST_STRING_LENGTH_ERR	3011	X'00000BC3'
MQRCCF_FORCE_VALUE_ERROR	3012	X'00000BC4'
MQRCCF_STRUCTURE_TYPE_ERROR	3013	X'00000BC5'
MQRCCF_CFIN_PARM_ID_ERROR	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_ERROR	3015	X'00000BC7'
MQRCCF_MSG_LENGTH_ERROR	3016	X'00000BC8'
MQRCCF_CFIN_DUPLICATE_PARM	3017	X'00000BC9'
MQRCCF_CFST_DUPLICATE_PARM	3018	X'00000BCA'
MQRCCF_PARM_COUNT_TOO_SMALL	3019	X'00000BCB'
MQRCCF_PARM_COUNT_TOO_BIG	3020	X'00000BCC'
MQRCCF_Q_ALREADY_IN_CELL	3021	X'00000BCD'
MQRCCF_Q_TYPE_ERROR	3022	X'00000BCE'
MQRCCF_MD_FORMAT_ERROR	3023	X'00000BCF'
MQRCCF_CFSL_LENGTH_ERROR	3024	X'00000BD0'
MQRCCF_REPLACE_VALUE_ERROR	3025	X'00000BD1'
MQRCCF_CFIL_DUPLICATE_VALUE	3026	X'00000BD2'
MQRCCF_CFIL_COUNT_ERROR	3027	X'00000BD3'
MQRCCF_CFIL_LENGTH_ERROR	3028	X'00000BD4'
MQRCCF_QUIESCE_VALUE_ERROR	3029	X'00000BD5'
MQRCCF_MODE_VALOR_ERROR	3029	X'00000BD5'
MQRCCF_MSG_SEQ_NUMBER_ERROR	3030	X'00000BD6'
MQRCCF_PING_DATA_COUNT_ERROR	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
MQRCCF_CHANNEL_TYPE_ERROR	3034	X'00000BDA'
MQRCCF_PARM_SEQUENCE_ERROR	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
MQRCCF_BATCH_SIZE_ERROR	3037	X'00000BDD'
MQRCCF_DISC_INT_ERROR	3038	X'00000BDE'
MQRCCF_SHORT_RETRY_ERROR	3039	X'00000BDF'
MQRCCF_SHORT_TIMER_ERROR	3040	X'00000BE0'
MQRCCF_LONG_RETRY_ERROR	3041	X'00000BE1'
MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'
MQRCCF_SEQ_NUMBER_WRAP_ERROR	3043	X'00000BE3'
MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_PUT_AUTH_ERROR	3045	X'00000BE5'
MQRCCF_PURGE_VALUE_ERROR	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_TRUNCATED	3048	X'00000BE8'
MQRCCF_CCSID_ERROR	3049	X'00000BE9'
MQRCCF_ENCODING_ERROR	3050	X'00000BEA'
MQRCCF_QUEUES_VALUE_ERROR	3051	X'00000BEB'
MQRCCF_DATA_CONV_VALUE_ERROR	3052	X'00000BEC'
MQRCCF_INDOUBT_VALUE_ERROR	3053	X'00000BED'
MQRCCF_ESCAPE_TYPE_ERROR	3054	X'00000BEE'
MQRCCF_REPOS_VALUE_ERROR	3055	X'00000BEF'
MQRCCF_CHANNEL_TABLE_ERROR	3062	X'00000BF6'
MQRCCF_MCA_TYPE_ERROR	3063	X'00000BF7'
MQRCCF_CHL_INST_TYPE_ERROR	3064	X'00000BF8'
MQRCCF_CHL_STATUS_NOT_FOUND	3065	X'00000BF9'
MQRCCF_CFSL_DUPLICATE_PARM	3066	X'00000BFA'
MQRCCF_CFSL_TOTAL_LENGTH_ERROR	3067	X'00000BFB'
MQRCCF_CFSL_COUNT_ERROR	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_DELETED	3070	X'00000BFE'
MQRCCF_STREAM_ERROR	3071	X'00000BFF'
MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NO_REGISTRADO	3073	X'00000C01'
MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_INCORRECT_STREAM	3075	X'00000C03'
MQRCCF_Q_NAME_ERROR	3076	X'00000C04'
MQRCCF_NO_RETAINED_MSG	3077	X'00000C05'
MQRCCF_DUPLICATE_IDENTITY	3078	X'00000C06'
MQRCCF_INCORRECT_Q	3079	X'00000C07'
MQRCCF_ID_CORREL_ERROR	3080	X'00000C08'
MQRCCF_NO_AUTORIZADO	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'
MQRCCF_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR	3084	X'00000C0C'
MQRCCF_UNKNOWN_BROKER	3085	X'00000C0D'
MQRCCF_Q_MGR_CCSID_ERROR	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'
MQRCCF_CLUSTER_NAME_CONFLICTO	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICTO	3089	X'00000C11'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ACTION_VALUE_ERROR	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_NETBIOS_NAME_ERROR	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID	3096	X'00000C18'
MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'
MQRCCF_USUARIO_INCORRECTO	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
MQRCCF_SUB_IDENTITY_ERROR	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED	3156	X'00000C54'
MQRCCF_ALREADY_JOINED	3157	X'00000C55'
MQRCCF_OBJECT_IN_USE	3160	X'00000C58'
MQRCCF_UNKNOWN_FILE_NAME	3161	X'00000C59'
MQRCCF_FILE_NOT_AVAILABLE	3162	X'00000C5A'
MQRCCF_DISC_RETRY_ERROR	3163	X'00000C5B'
MQRCCF_ALLOC_RETRY_ERROR	3164	X'00000C5C'
MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
MQRCCF_NÚMERO_PUERTO_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ACTIVE	3168	X'00000C60'
MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
MQRCCF_PROFILE_NAME_ERROR	3170	X'00000C62'
MQRCCF_AUTH_VALUE_ERROR	3171	X'00000C63'
MQRCCF_AUTH_VALUE_MISSING	3172	X'00000C64'
MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
MQRCCF_CONNECTION_ID_ERROR	3174	X'00000C66'
MQRCCF_LOG_TYPE_ERROR	3175	X'00000C67'
MQRCCF_PROGRAM_NOT_AVAILABLE	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_FAILED	3177	X'00000C69'
MQRCCF_NONE_FOUND	3200	X'00000C80'
MQRCCF_SECURITY_SWITCH_OFF	3201	X'00000C81'
MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'
MQRCCF_PARM_CONFLICT	3203	X'00000C83'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_COMMAND_INHIBIDO	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_NOMBRE_OBJETO_RESTRINGIDO	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_SUPERADO	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICTO	3211	X'00000C8B'
MQRCCF_Q_MGR_NOT_IN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
MQRCCF_NAMELIST_ERROR	3215	X'00000C8F'
MQRCCF_NO_CHANNEL_INITIATOR	3217	X'00000C91'
MQRCCF_CHANNEL_INITIATOR_ERROR	3218	X'00000C92'
MQRCCF_COMMAND_LEVEL_CONFLICTO	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICTO	3223	X'00000C97'
MQRCCF_EVENTS_DISABLED	3224	X'00000C98'
MQRCCF_COMMAND_SCOPE_ERROR	3225	X'00000C99'
MQRCCF_COMMAND_REPLY_ERROR	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED	3227	X'00000C9B'
MQRCCF_PARM_MISSING	3228	X'00000C9C'
MQRCCF_PARM_VALUE_ERROR	3229	X'00000C9D'
MQRCCF_COMMAND_LENGTH_ERROR	3230	X'00000C9E'
MQRCCF_COMMAND_ORIGIN_ERROR	3231	X'00000C9F'
MQRCCF_LISTENER_CONFLICTO	3232	X'00000CA0'
MQRCCF_LISTENER_STARTED	3233	X'00000CA1'
MQRCCF_LISTENER_STOPPED	3234	X'00000CA2'
MQRCCF_CHANNEL_ERROR	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR	3236	X'00000CA4'
MQRCCF_UNKNOWN_USER_ID	3237	X'00000CA5'
MQRCCF_UNEXPECTED_ERROR	3238	X'00000CA6'
MQRCCF_NO_XCF_PARTNER	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_ERROR	3240	X'00000CA8'
MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
MQRCCF_CFIF_OPERATOR_ERROR	3242	X'00000CAA'
MQRCCF_CFIF_PARM_ID_ERROR	3243	X'00000CAB'
MQRCCF_CFSF_FILTER_VAL_LEN_ERR	3244	X'00000CAC'
MQRCCF_CFSF_LENGTH_ERROR	3245	X'00000CAD'
MQRCCF_CFSF_OPERATOR_ERROR	3246	X'00000CAE'
MQRCCF_CFSF_PARM_ID_ERROR	3247	X'00000CAF'
MQRCCF_TOO_MANY_FILTERS	3248	X'00000CB0'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_LISTENER_RUNNING	3249	X'00000CB1'
MQRCCF_LSTR_STATUS_NOT_FOUND	3250	X'00000CB2'
MQRCCF_SERVICE_RUNNING	3251	X'00000CB3'
MQRCCF_SERV_STATUS_NOT_FOUND	3252	X'00000CB4'
MQRCCF_SERVICE_STOPPED	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
MQRCCF_CFBS_LENGTH_ERROR	3255	X'00000CB7'
MQRCCF_CFBS_PARM_ID_ERROR	3256	X'00000CB8'
MQRCCF_CFBS_STRING_LENGTH_ERR	3257	X'00000CB9'
MQRCCF_CFGR_LENGTH_ERROR	3258	X'00000CBA'
MQRCCF_CFGR_PARM_COUNT_ERROR	3259	X'00000CBB'
MQRCCF_CONN_NOT_STOPPED	3260	X'00000CBC'
MQRCCF_SERVICE_REQUEST_PENDING	3261	X'00000CBD'
MQRCCF_NO_START_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
MQRCCF_CFBF_LENGTH_ERROR	3264	X'00000CC0'
MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
MQRCCF_CFBF_OPERATOR_ERROR	3266	X'00000CC2'
MQRCCF_CFBF_FILTER_VAL_LEN_ERR	3267	X'00000CC3'
MQRCCF_LISTENER_STILL_ACTIVE	3268	X'00000CC4'
MQRCCF_DEF_XMIT_Q_CLUS_ERROR	3269	X'0000CC5'
MQRCCF_TOPICSTR_ALREADY_EXISTS	3300	X'00000CE4'
MQRCCF_SHARING_CONVS_ERROR	3301	X'00000CE5'
MQRCCF_SHARING_CONVS_TYPE	3302	X'00000CE6'
MQRCCF_SECURITY_CASE_CONFLICTO	3303	X'00000CE7'
MQRCCF_TOPIC_TYPE_ERROR	3305	X'00000CE9'
MQRCCF_MAX_INSTANCES_ERROR	3306	X'00000CEA'
MQRCCF_MAX_INSTS_PER_CLNT_ERR	3307	X'00000CEB'
MQRCCF_TOPIC_STRING_NOT_FOUND	3308	X'00000CEC'
MQRCCF_SUBSCRIPTION_POINT_ERR	3309	X'00000CED'
MQRCCF_SUB_ALREADY_EXISTS	3311	X'00000CEF'
MQRCCF_UNKNOWN_OBJECT_NAME	3312	X'00000CF0'
MQRCCF_REMOTE_Q_NAME_ERROR	3313	X'00000CF1'
MQRCCF_DURABILITY_NOT_ALLOWED	3314	X'00000CF2'
MQRCCF_HOBJ_ERROR	3315	X'00000CF3'
MQRCCF_DEST_NAME_ERROR	3316	X'00000CF4'
MQRCCF_INVALID_DESTINATION	3317	X'00000CF5'
MQRCCF_PUBSUB_XX_ENCODE_CASE_ONE inhibida	3318	X'00000CF6'
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_TIPO_AUTORIZACIÓN_ERRÓNEO	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_EQUIVOC_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_INJU_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_ERRÓNEA_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICTO	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_SUPERADO	3344	X'00000D10'
MQRCCF_IPADDR_ERROR	3345	X'00000D11'
MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_PROFILE_NAME_MISSING	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_INVALID_PROTOCOL	3365	X'00000D25'
V 9.1.0 V 9.1.0 MQRCCF_ACCESS_BLOCKED	3382	X'00000D36'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_TIPO_INCORRECTO	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_TIPO_INCORRECTO	4003	X'00000FA3'
MQRCCF_OBJECT_OPEN	4004	X'00000FA4'
MQRCCF_ATTR_VALUE_ERROR	4005	X'00000FA5'
MQRCCF_UNKNOWN_Q_MGR	4006	X'00000FA6'
MQRCCF_Q_TIPO_INCORRECTO	4007	X'00000FA7'
MQRCCF_OBJECT_NAME_ERROR	4008	X'00000FA8'
MQRCCF_ALLOCATE_FAILED	4009	X'00000FA9'
MQRCCF_HOST_NOT_AVAILABLE	4010	X'00000FAA'
MQRCCF_CONFIGURATION_ERROR	4011	X'00000FAB'
MQRCCF_CONNECTION_RECHAZADO	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR	4013	X'00000FAD'
MQRCCF_SEND_FAILED	4014	X'00000FAE'
MQRCCF_RECEIVED_DATA_ERROR	4015	X'00000FAF'
MQRCCF_RECEIVE_FAILED	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'
MQRCCF_NO_STORAGE	4018	X'00000FB2'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED	4020	X'00000FB4'
MQRCCF_BIND_FAILED	4024	X'00000FB8'
MQRCCF_CHANNEL_INDOUBT	4025	X'00000FB9'
MQRCCF_MQCONN_FAILED	4026	X'00000FBA'
MQRCCF_MQOPEN_FAILED	4027	X'00000FBB'
MQRCCF_MQGET_FAILED	4028	X'00000FBC'
MQRCCF_MQPUT_FAILED	4029	X'00000FBD'
MQRCCF_ERROR	4030	X'00000FBE'
MQRCCF_CHANNEL_IN_USE	4031	X'00000FBF'
MQRCCF_CHANNEL_NOT_FOUND	4032	X'00000FC0'
MQRCCF_UNKNOWN_REMOTE_CHANNEL	4033	X'00000FC1'
MQRCCF_REMOTE_QM_UNAVAILABLE	4034	X'00000FC2'
MQRCCF_REMOTE_QM_TERMINANDO	4035	X'00000FC3'
MQRCCF_MQINQ_FAILED	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NOT_AVAILABLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_TIPO_CANAL_INCORRECTO	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
MQRCCF_CHANNEL_NAME_ERROR	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
MQRCCF_MCA_NAME_ERROR	4047	X'00000FCF'
MQRCCF_SEND_EXIT_NAME_ERROR	4048	X'00000FD0'
MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
MQRCCF_MSG_EXIT_NAME_ERROR	4050	X'00000FD2'
MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_INJUSTA_TYPE	4052	X'00000FD4'
MQRCCF_MCA_NAME_INJUSTA_TYPE	4053	X'00000FD5'
MQRCCF_DISC_INT_TIPO_INCORRECTO	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_INJU_TYPE	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_INJU_TYPE	4056	X'00000FD8'
MQRCCF_LONG_RETRY_INJU_TYPE	4057	X'00000FD9'
MQRCCF_LONG_TIMER_INJUSTA_TYPE	4058	X'00000FDA'
MQRCCF_PUT_AUTH_TIPO_ERRÓNEO	4059	X'00000FDB'
MQRCCF_KEEP_ALIVE_INT_ERROR	4060	X'00000FDC'
MQRCCF_MISSING_CONN_NAME	4061	X'00000FDD'

Tabla 307. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_FAILED	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ACTIVE	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NO_DISPONIBLE	4068	X'00000FE4'
MQRCCF_MR_COUNT_ERROR	4069	X'00000FE5'
MQRCCF_MR_COUNT_WRONG_TYPE	4070	X'00000FE6'
MQRCCF_MR_EXIT_NAME_ERROR	4071	X'00000FE7'
MQRCCF_MR_EXIT_NAME_INJUSTA_TYPE	4072	X'00000FE8'
MQRCCF_MR_INTERVAL_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVAL_TIPO_INCORRECTO	4074	X'00000FEA'
MQRCCF_NPM_SPEED_ERROR	4075	X'00000FEB'
MQRCCF_NPM_SPEED_WRONG_TYPE	4076	X'00000FEC'
MQRCCF_HB_INTERVAL_ERROR	4077	X'00000FED'
MQRCCF_HB_INTERVAL_TIPO_ERRÓNEO	4078	X'00000FEE'
MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_TIPO_INCORRECTO	4080	X'00000FF0'
MQRCCF_CHAD_EVENT_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENT_INJU_TYPE	4082	X'00000FF2'
MQRCCF_CHAD_EXIT_ERROR	4083	X'00000FF3'
MQRCCF_CHAD_EXIT_TIPO_INCORRECTO	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
MQRCCF_BATCH_INT_ERROR	4086	X'00000FF6'
MQRCCF_BATCH_INT_TIPO_INCORRECTO	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_TIPO_INCORRECTO	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED	4095	X'00000FFF'
z/OS V 9.1.3 MQRCCF_KWD_VALUE_WRONG_TYPE	4096	X'00001000'

MQRCN_ * (Constantes de reconexión de cliente)

Tabla 308. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_ * (tipos de tiempo de espera de recepción)

Tabla 309. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_ * (Leer valores de cabecera)

Tabla 310. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREADA_NO	0	X'00000000'
MQREADA_SÍ	1	X'00000001'
MQREADA_DISABLED	2	X'00000002'
MQREADA_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	3	X'00000003'
MQREADA_BACKLOG	4	X'00000004'

MQRECORDING_ * (Opciones de grabación)

Tabla 311. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRECORDING_INHABILITADO	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_ * (Opciones de registro de publicación/suscripción)

Tabla 312. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREGO_NONE	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQREGO_ANONYMOUS	2	X'00000002'
MQREGO_LOCAL	4	X'00000004'
MQREGO_DIRECT_REQUESTS	8	X'00000008'
MQREGO_NEW_PUBLICATIONS_ONLY	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY	32	X'00000020'

Tabla 312. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_STREAM_NAME	128	X'00000080'
MQREGO_INFORM_IF_RETAINED	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'
MQREGO_NON_PERSISTENT	1024	X'00000400'
MQREGO_PERSISTENT	2048	X'00000800'
MQREGO_PERSISTENT_AS_PUBLISH	4096	X'00001000'
MQREGO_PERSISTENT_AS_Q	8192	X'00002000'
MQREGO_ADD_NAME	16384	X'00004000'
MQREGO_NO_ALTERACIÓN	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY	524288	X'00080000'
MQREGO_VARIABLE_USER_ID	1048576	X'00100000'
MQREGO_LOCKED	2097152	X'00200000'

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

Reglas y estructura de cabecera de formato

Tabla 313. Estructuras de constantes	
Nombre	**Estructura**
MQRFH_STRUC_ID	"RFH¬"
MQRFH_STRUC_ID_ARRAY	'R','F','H','¬'

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

Tabla 314. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRFH_VERSION_1	1	X'00000001'
MQRFH_VERSION_2	2	X'00000002'
MQRFH_STRUC_LENGTH_FIXED	32	X'00000020'
MQRFH_STRUC_LENGTH_FIXED_2	36	X'00000024'

Reglas y distintivos de cabecera de formato

Tabla 315. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

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

Tabla 316. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

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

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

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

MQRFH2_PUBSUB_CMD_FOLDER_B	"<psc>"
MQRFH2_PUBSUB_CMD_FOLDER_E	"</psc>"
MQRFH2_PUBSUB_RESP_FOLDER_B	"<pscr>"
MQRFH2_PUBSUB_RESP_FOLDER_E	"</pscr>"
MQRFH2_MSG_CONTENT_FOLDER_B	"<mcd>"
MQRFH2_MSG_CONTENT_FOLDER_E	"</mcd>"
MQRFH2_USER_FOLDER_B	"<usr>"
MQRFH2_USER_FOLDER_E	"</usr>"

MQRL_ * (Longitud devuelta)

Tabla 317. Valores de constantes

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

MQRMH_ * (estructura de cabecera de mensaje de referencia)

Tabla 318. Estructuras de constantes

Nombre	Estructura
MQRMH_STRUC_ID	"RMH¬"
MQRMH_STRUC_ID_ARRAY	'R','M','H','¬'

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

Tabla 319. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRMH_VERSION_1	1	X'00000001'
MQRMH_CURRENT_VERSION	1	X'00000001'

MQRMHF_ * (Distintivos de cabecera de mensaje de referencia)

Tabla 320. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRMHF_LAST	1	X'00000001'
MQRMHF_NOT_LAST	0	X'00000000'

MQRO_ * (Opciones de informe)

Tabla 321. Valores de constantes

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

*MQRO_ * (máscaras de opciones de informe)*

Tabla 322. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRO_REJECT_UNSUP_MASK	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_ * (ruta de rastreo)

Máximo de actividades de ruta de rastreo (MQIACF_MAX_ACTIVITIES)

Tabla 323. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Detalle de ruta de rastreo (MQIACF_ROUTE_DETAIL)

Tabla 324. Valores de constantes

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

Reenvío de ruta de rastreo (MQIACF_ROUTE_FORWARDING)

Tabla 325. Valores de constantes

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

Entrega de ruta de rastreo (MQIACF_ROUTE_DELIVERY)

Tabla 326. Valores de constantes

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

Acumulación de ruta de rastreo (MQIACF_ROUTE_ACUMULACIÓN)

Tabla 327. Valores de constantes

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

MQRP_ * (Formato de mandato, Opciones de sustitución)

Tabla 328. Valores de constantes

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

MQRQ_ * (Formato de mandato, calificadores de razón)

Tabla 329. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRQ_CONN_NOT_AUTHORIZED	1	X'00000001'
MQRQ_OPEN_NOT_AUTHORIZED	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED	4	X'00000004'
MQRQ_Q_MGR_DETENIENDO	5	X'00000005'
MQRQ_Q_MGR_QUIESCING	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_BRIDGE_STOPPED_OK	11	X'0000000B'
MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'
MQRQ_SYS_CONN_NOT_AUTHORIZED	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_NO_INSTALADO	28	X'0000001C'

MQRT_ * (Formato de mandato, Tipos de renovación)

Tabla 330. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
CONFIGURACIÓN_MQR	1	X'00000001'
MQRT_CADUCIDAD	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_ * (sólo solicitud)

Tabla 331. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_ * (autenticación de cliente TLS)

Tabla 332. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCA_REQUIRED	0	X'00000000'
MQSCA_OPTIONAL	1	X'00000001'

MQSCO_ * (opciones de configuración de TLS)

Estructura de opciones de configuración TLS

Tabla 333. Estructuras de constantes	
Nombre	**Estructura**
MQSCO_ID_ESTRUCTURA	"SCO¬"
MQSCO_STRUC_ID_ARRAY	'S','C','O','¬'

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

Tabla 334. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
MQSCO_VERSIÓN_ACTUAL	4	X'00000004'

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

Recuento de restablecimiento de claves de opciones de configuración TLS

Tabla 335. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Formato de mandato de ámbito de definición de cola

Tabla 336. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_ * (ámbito de publicación)

Tabla 337. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCOPE_TODOS	0	X'00000000'
MQSCOPE_AS_PARENT	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_ * (Caso de seguridad)

Tabla 338. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCYC_UPPER	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_ * (Estructura de descriptor de objeto)

Tabla 339. Nombres y estructuras de constantes	
Nombre	**Estructura**
MQSD_STRUC_ID	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

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

Tabla 340. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSD_VERSION_1	1	X'00000001'
MQSD_CURRENT_VERSION	1	X'00000001'

MQSECITEM_ * (Formato de mandato, Elementos de seguridad)

Tabla 341. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECITEM_ALL	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMDS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECPROT_ * (Tipos de protocolo de seguridad)

Tabla 342. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECPROT_NONE	0	X'00000000'
MQSECPROT_SSLV30	1	X'00000001'
MQSECPROT_TLSV10	2	X'00000002'
MQSECPROT_TLSV12	4	X'00000004'

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

Formato de mandato Conmutadores de seguridad

Tabla 343. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
PROCESO DE MQSECSW	1	X'00000001'
MQSECSW_NAMELIST	2	X'00000002'
MQSECSW_Q	3	X'00000003'
MQSECSW_TOPIC	4	X'00000004'
CONTEXTO_MQSECSW_CONTEXTO	6	X'00000006'
MQSECSW_ALTERNATE_USER	7	X'00000007'
MQSECSW_COMMAND	8	X'00000008'
MQSECSW_CONNECTION	9	X'00000009'
MQSECSW_SUBSYSTEM	10	X'0000000A'
MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Formato de mandato Estado de conmutador de seguridad

Tabla 344. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECSW_OFF_FOUND	21	X'00000015'
MQSECSW_ON_FOUND	22	X'00000016'
MQSECSW_OFF_NOT_FOUND	23	X'00000017'
MQSECSW_ON_NOT_FOUND	24	X'00000018'
MQSECSW_OFF_ERROR	25	X'00000019'
MQSECSW_ON_OVERRIDDEN	26	X'0000001A'

MQSECTYPE_ * (Formato de mandato, Tipos de seguridad)

Tabla 345. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECTYPE_AUTHSERV	1	X'00000001'

Tabla 345. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECTYPE_SSL	2	X'00000002'
MQSECTYPE_CLASSES	3	X'00000003'

MQSEG_ * (Segmentación)

Tabla 346. Nombres y valores de constantes	
Nombre	**Valor**
MQSEG_INHIBIDO	'¬'
MQSEG_PERMITIDO	'A'

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

MQSEL_ * (Valores de selector especiales)

Tabla 347. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_ANY_USER_SELECTOR	-30002	X'FFFF8ACE'
MQSEL_ANY_SYSTEM_SELECTOR	-30003	X'FFFF8ACD'
MQSEL_ALL_SELECTORS	-30001	X'FFFF8ACF'
MQSEL_ALL_USER_SELECTORS	-30002	X'FFFF8ACE'
MQSEL_ALL_SYSTEM_SELECTORS	-30003	X'FFFF8ACD'

MQSELTYPE_ * (Tipos de selector)

Tabla 348. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED	2	X'00000002'

MQSID_ * (Identificador de seguridad)

Tabla 349. Nombres y valores de constantes	
Nombre	**Valor**
MQSID_NONE	X'00...00' (40 nulos)
MQSID_NONE_ARRAY	'\0','\0',... (40 nulos)

MQSIDT_ * (Tipos de identificador de seguridad)

Tabla 350. Nombres y valores de constantes	
Nombre	**Valor hexadecimal**
MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

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

Establecer estructura de opciones de propiedad de mensaje

Tabla 351. Estructuras de constantes

Nombre	Estructura
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

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

Tabla 352. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_VERSIÓN_ACTUAL	1	X'00000001'

Establecer opciones de propiedad de mensaje

Tabla 353. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSMPO_SET_FIRST	0	X'00000000'
MQSMPO_SET_PROP_UNDER_CURSOR	1	X'00000001'
MQSMPO_SET_PROP_AFTER_CURSOR	2	X'00000002'
MQSMPO_APPEND_PROPERTY	4	X'00000004'
MQSMPO_SET_PROP_BEFORE_CURSOR	8	X'00000008'
MQSMPO_NONE	0	X'00000000'

MQSO_ * (Opciones de suscripción)

Tabla 354. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSO_NONE	0	X'00000000'
MQSO_NON_DURABLE	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQSO_ALTER	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE	8	X'00000008'
SUB_GRUPO_MQSO	16	X'00000010'
MQSO_GESTIONADO	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLICATIONS_ONLY	4096	X'00001000'
MQSO_FAIL_IF_QUIESCING	8192	X'00002000'

Tabla 354. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSO_ALTERNATE_USER_AUTHORITY	262144	X'00040000'
MQSO_WILDCARD_CHAR	1048576	X'00100000'
MQSO_WILDCARD_TOPIC	2097152	X'00200000'
MQSO_SET_CORREL_ID	4194304	X'00400000'
MQSO_SCOPE_QMGR	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_ * (Disponibilidad de punto de sincronización)

Tabla 355. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

V 9.1.3 MQSPL_ * (Opciones de protección de política de seguridad)

Tabla 356. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSPL_PASSTHRU	0	X'00000000'
MQSPL_ELIMINAR	1	X'00000001'
MQSPL_AS_POLICY	2	X'00000002'

MQSQQM_ * (Nombre de gestor de colas compartido)

Tabla 357. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSQQM_USE	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_ * (Acción)

Tabla 358. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSR_ACTION_PUBLICATION	1	X'00000001'

MQSRO_ * (estructura de opciones de solicitud de suscripción)

Tabla 359. Estructuras de constantes	
Nombre	**Estructura**
MQSRO_STRUC_ID	"SRO¬"
MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

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

Tabla 360. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSRO_VERSION_1	1	X'00000001'
MQSRO_CURRENT_VERSION	1	X'00000001'
MQSRO_NONE	0	X'00000000'
MQSRO_FAIL_IF_QUIESCING	8192	X'00002000'

MQSS_ * (Estado de segmento)

Tabla 361. Nombres y estructuras de constantes	
Nombre	**Estructura**
MQSS_NO_A_SEGMENTO	'¬'
SEGMENTO_MQSS	'S'
MQSS_LAST_SEGMENT	'L'

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

MQSSL_ * (Requisitos de TLS FIPS)

Tabla 362. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_ * (Opciones de estadísticas)

Tabla 363. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'
MQSTAT_TYPE_RECONNECTION	0	X'00000000'
MQSTAT_TYPE_RECONNECTION_ERROR	0	X'00000000'

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

Tabla 364. Estructuras de constantes	
Nombre	**Estructura**
MQSTS_ID_STRUCT	"STAT"
MQSTS_MATRIZ_ID_ESTRUCTURA	'S','T','A','T'

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

Tabla 365. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSTS_VERSION_1	1	X'00000001'
MQSTS_CURRENT_VERSION	1	X'00000001'

MQSUB_ * (Suscripciones duraderas)

Suscripciones duraderas

Tabla 366. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBIDO	2	X'00000002'

Suscripciones duraderas

Tabla 367. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_ * (Formato de mandato, Tipos de suscripción)

Tabla 368. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSUBTYPE_API	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
PROXY MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
MQSUBTYPE_USER	-2	X'FFFFFFFE'

MQSUS_ * (Formato de mandato, Estado de suspensión)

Tabla 369. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSUS_SÍ	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_ * (Servicio)

Tipos de servicio

Tabla 370. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSVC_TYPE_COMMAND	0	X'00000000'
MQSVC_TYPE_SERVER	1	X'00000001'

Controles de servicio

Tabla 371. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL	2	X'00000002'

Estado del servicio

Tabla 372. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSVC_STATUS_STOPPED	0	X'00000000'
MQSVC_STATUS_STARTING	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
MQSVC_STATUS_DETENIÉNDOSE	3	X'00000003'
MQSVC_STATUS_REINTENTANDO	4	X'00000004'

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

Tabla 373. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSYNCPOINT_YES	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

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

Tabla 374. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSYSP_NO	0	X'00000000'
MQSYSP_SÍ	1	X'00000001'
MQSYSP_AMPLIADO	2	X'00000002'
MQSYSP_TIPO_INICIAL	10	X'0000000A'
MQSYSP_TYPE_SET	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS	13	X'0000000D'
MQSYSP_TYPE_ARCHIVE_TAPE	14	X'0000000E'
MQSYSP_ALLOC_BLK	20	X'00000014'
MQSYSP_ALLOC_TRK	21	X'00000015'
MQSYSP_ALLOC_CYL	22	X'00000016'
MQSYSP_STATUS_OCUPADO	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_AVAILABLE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'

Tabla 374. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
MQSYSP_STATUS_COPYING_LOG	36	X'00000024'

MQTA_ * (Atributos de tema)

Comodines

Tabla 375. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

Suscripciones permitidas

Tabla 376. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_XX_ENCODE_CASE_ONE inhibida	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

Subpropagación de proxy

Tabla 377. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Publicaciones permitidas

Tabla 378. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_XX_ENCODE_CASE_ONE inhibida	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_ * (Controles desencadenantes)

Tabla 379. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_ * (TCP Keepalive)

Tabla 380. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_ * (Tipos de pila TCP)

Tabla 381. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_ * (Formato de mandato, Unidades de tiempo)

Tabla 382. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_ * (Desencadenar estructura de mensaje)

Tabla 383. Estructuras de constantes	
Nombre	**Estructura**
MQTM_STRUC_ID	"TM¬¬"
MQTM_STRUC_ID_ARRAY	'T','M','¬','¬'

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

Tabla 384. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

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

Tabla 385. Estructuras de constantes	
Nombre	**Estructura**
MQTMC_STRUC_ID	"TMC¬"
MQTMC_STRUC_ID_ARRAY	'T','M','C','¬'
MQTMC_VERSION_1	"¬¬¬1"
MQTMC_VERSION_2	"¬¬¬2"
MQTMC_CURRENT_VERSION	"¬¬¬2"
MQTMC_VERSION_1_ARRAY	'¬','¬','¬','1'
MQTMC_VERSION_2_ARRAY	'¬','¬','¬','2'
MQTMC_CURRENT_VERSION_ARRAY	'¬','¬','¬','2'

MQTOPT_ * (Tipo de tema)

Tabla 386. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTOPT_LOCAL	0	X'00000000'
MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

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

Tabla 387. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_SÍ	1	X'00000001'

MQTSCOPE_ * (Ámbito de suscripción)

Tabla 388. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_ * (Tipos de desencadenante)

Tabla 389. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_ * (Tipos de datos de propiedad)

Tabla 390. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
MQTYPE_BOOLEAN	4	X'00000004'
MQTYPE_BYTE_STRING	8	X'00000008'
MQTYPE_INT8	16	X'00000010'
MQTYPE_INT16	32	X'00000020'
MQTYPE_INT32	64	X'00000040'
MQTYPE_LONG	64	X'00000040'
MQTYPE_INT64	128	X'00000080'
MQTYPE_FLOAT32	256	X'00000100'
MQTYPE_FLOAT64	512	X'00000200'

Tabla 390. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTYPE_STRING	1024	X'00000400'

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

Tabla 391. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUA_PRIMERO	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

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

Tabla 392. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_SÍ	1	X'00000001'

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

Tabla 393. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_ * (Formato de mandato, Estados de UOW)

Tabla 394. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUOWST_NONE	0	X'00000000'
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARADO	2	X'00000002'
MQUOWST_SIN resolver	3	X'00000003'

MQUOWT_ * (Formato de mandato, Tipos de UOW)

Tabla 395. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
MQUOWT_IMS	3	X'00000003'
MQUOWT_XA	4	X'00000004'

MQUS_ * (Usos de cola)

Tabla 396. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION	1	X'00000001'

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

Formato de mandato Valores de uso de conjunto de páginas

Tabla 397. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUSAGE_PS_AVAILABLE	0	X'00000000'
MQUSAGE_PS_DEFINED	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_PS_SUSPENDED	4	X'00000004'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SISTEMA	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Formato de mandato Valores de uso de conjunto de datos

Tabla 398. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUSAGE_DS_OLDEST_ACTIVE_UOW	10	X'0000000A'
MQUSAGE_DS_OLDEST_PS_RECOVERY	11	X'0000000B'
MQUSAGE_DS_OLDEST_CF_RECOVERY	12	X'0000000C'

MQVL_ * (Longitud de valor)

Tabla 399. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQVL_NULL_TERMINATED	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_ * (ID de usuario variable)

Tabla 400. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
USUARIO_ARREGLADO_MQVU_USUARIO	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

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

Tabla 401. Estructuras de constantes	
Nombre	**Estructura**
MQWDR_STRUC_ID	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

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

Tabla 402. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
MQWDR_CURRENT_VERSION	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
MQWDR_CURRENT_LENGTH	136	X'00000088'

MQWI_ * (Intervalo de espera)

Tabla 403. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWI_UNLIMITED	-1	X'FFFFFFFF'

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

Estructura de cabecera de información de carga de trabajo

Tabla 404. Estructuras de constantes	
Nombre	**Estructura**
MQWIH_STRUC_ID	"WIH¬"
MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

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

Tabla 405. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

Distintivos de cabecera de información de carga de trabajo

Tabla 406. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWIH_NONE	0	X'00000000'

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

Tabla 407. Estructuras de constantes	
Nombre	**Estructura**
MQWQR_STRUC_ID	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

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

Tabla 408. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
MQWQR_CURRENT_VERSION	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_CURRENT_LENGTH	212	X'000000D4'

MQWS_ * (Esquema comodín)

Tabla 409. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWS_PREDETERMINADO	0	X'00000000'
MQWS_CHAR	1	X'00000001'
MQWS_TOPIC	2	X'00000002'

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

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

Tabla 410. Estructuras de constantes	
Nombre	**Estructura**
MQWXP_STRUC_ID	"WXP¬"
MQWXP_STRUC_ID_ARRAY	'W','X','P','¬'

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

Tabla 411. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
MQWXP_CURRENT_VERSION	4	X'00000004'

MQWXP_ * (distintivos de carga de trabajo de clúster)

Tabla 412. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Referencia relacionada

"Campos en MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster" en la página 1584
Descripción de los campos en MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster

MQXACT_ * (Tipos de interlocutor de API)

Tabla 413. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_ * (mandatos de salida)

Tabla 414. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXC_MQOPEN	1	X'00000001'
MQXC_MQCLOSE	2	X'00000002'
MQXC_MQGET	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQXC_MQINQ	6	X'00000006'
MQXC_MQSET	8	X'00000008'
MQXC_MQBACK	9	X'00000009'
MQXC_MQCMIT	10	X'0000000A'

MQXCC_ * (Respuestas de salida)

Tabla 415. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXCC_Correcto	0	X'00000000'
MQXCC_SUPPRESS_FUNCTION	-1	X'FFFFFFFF'
MQXCC_SKIP_FUNCTION	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'

Tabla 415. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
MQXCC_FAILED	-8	X'FFFFFFF8'

MQXDR_ * (Respuesta de salida)

Tabla 416. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXDR_Correcto	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_ * (Entornos)

Tabla 417. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

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

Estructura de opciones de punto de entrada de registro

Tabla 418. Estructuras de constantes	
Nombre	**Estructura**
MQXEPO_ID_ESTRUCTURA	"XEPO"
MQXEPO_STRUC_ID_ARRAY	'X','E','P','O'

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

Tabla 419. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_VERSIÓN_ACTUAL	1	X'00000001'

Opciones de salida

Tabla 420. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXEPO_NONE	0	X'00000000'

MQXF_ * (Identificadores de función de API)

Tabla 421. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DISC	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_CLOSE	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_GET	10	X'0000000A'
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_INICIO	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_BACK	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMPLETE	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

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

Tabla 422. Estructuras de constantes	
Nombre	**Estructura**
MQXP_STRUC_ID	"XP¬¬"
MQXP_STRUC_ID_ARRAY	'X','P','¬','¬'

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

Tabla 423. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXP_VERSION_1	1	X'00000001'

MQXPDA_ * (Área de determinación de problemas)

Tabla 424. Nombres y valores de constantes	
Nombre	**Valor**
MQXPDA_NONE	X'00...00' (48 nulos)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 nulos)

MQXPT_ * (Tipos de transporte)

Tabla 425. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXPT_TODOS	-1	X'FFFFFFFF'
MQXPT_LOCAL	0	X'00000000'
MQXPT_LU62	1	X'00000001'
MQXPT_TCP	2	X'00000002'
MQXPT_NETBIOS	3	X'00000003'
MQXPT_SPX	4	X'00000004'
MQXPT_DECNET	5	X'00000005'
MQXPT_UDP	6	X'00000006'

MQXQH_ * (Estructura de cabecera de cola de transmisión)

Tabla 426. Estructuras de constantes	
Nombre	**Estructura**
MQXQH_STRUC_ID	"XQH¬"
MQXQH_STRUC_ID_ARRAY	'X','Q','H','¬'

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

Tabla 427. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_ * (Razones de salida)

Tabla 428. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXR_ANTES	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'
MQXR_INIT	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
MQXR_XMIT	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INIT_SEC	16	X'00000010'
MQXR_REINTENTAR	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
MQXR_ACK_RECEIVED	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'
MQXR_SEC_PARMS	29	X'0000001D'

MQXR2_* (Respuesta de salida 2)

Tabla 429. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'
MQXR2_USE_AGENT_BUFFER	0	X'00000000'
MQXR2_USE_EXIT_BUFFER	4	X'00000004'
MQXR2_DEFAULT_CONTINUATION	0	X'00000000'
MQXR2_CONTINUE_CHAIN	8	X'00000008'
MQXR2_SUPPRESS_CHAIN	16	X'00000010'
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_ * (Identificadores de salida)

Tabla 430. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXT_API_CROSSING_EXIT	1	X'00000001'
MQXT_API_EXIT	2	X'00000002'
MQXT_CHANNEL_SEC_EXIT	11	X'0000000B'
MQXT_CHANNEL_MSG_EXIT	12	X'0000000C'
MQXT_CHANNEL_SEND_EXIT	13	X'0000000D'
MQXT_CHANNEL_RCV_EXIT	14	X'0000000E'
MQXT_CHANNEL_MSG_RETRY_EXIT	15	X'0000000F'
MQXT_CHANNEL_AUTO_DEF_EXIT	16	X'00000010'
MQXT_CLUSTER_WORKLOAD_EXIT	20	X'00000014'
MQXT_PUBSUB_ROUTING_EXIT	21	X'00000015'

MQXUA_ * (Valor de área de usuario de salida)

Tabla 431. Nombres y valores de constantes	
Nombre	**Valor**
MQXUA_NONE	X'00...00' (16 nulos)
MQXUA_NOE_MATRIZ	'\0','\0',... (16 nulos)

MQXWD_ * (estructura del descriptor de espera de salida)

Tabla 432. Estructuras de constantes	
Nombre	**Estructura**
MQXWD_ID_STRUCD	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X','W','D','¬'

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

Tabla 433. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXWD_VERSION_1	1	X'00000001'

MQZAC_ * (Estructura de contexto de aplicación)

Tabla 434. Estructuras de constantes	
Nombre	**Estructura**
MQZAC_STRUC_ID	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z','A','C','¬'

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

Tabla 435. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_ * (estructura de datos de autorización)

Tabla 436. Estructuras de constantes	
Nombre	**Estructura**
MQZAD_STRUC_ID	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

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

Tabla 437. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_ * (Tipos de entidad de servicios instalables)

Tabla 438. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAET_NONE	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
GRUPO_MQZAC	2	X'00000002'
MQZAET_DESCONOCIDO	3	X'00000003'

MQZAO_ * (Autorizaciones de servicios instalables)

Tabla 439. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAO_CONNECT	1	X'00000001'
MQZAO_BROWSE	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_OUTPUT	8	X'00000008'
MQZAO_INQUIRE	16	X'00000010'
MQZAO_SET	32	X'00000020'
MQZAO_PASS_IDENTITY_CONTEXT	64	X'00000040'
MQZAO_PASS_ALL_CONTEXT	128	X'00000080'
MQZAO_SET_ IDENTITY_ CONTEXT	256	X'00000100'
MQZAO_SET_ ALL_CONTEXT	512	X'00000200'
MQZAO_ALTERNATE_USER_AUTHORITY	1024	X'00000400'
MQZAO_PUBLISH	2048	X'00000800'
MQZAO_SUBSCRIBE	4096	X'00001000'
MQZAO_RESUME	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
MQZAO_CREAR	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
MQZAO_DISPLAY	262144	X'00040000'

Tabla 439. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAO_CHANGE	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTORIZAR	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_TODOS	16662527	X'00FE3FFF'
MQZAO_ELIMINAR	16777216	X'01000000'
MQZAO_NONE	0	X'00000000'

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

Tabla 440. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAS_VERSION_1	1	X'00000001'
MQZAS_VERSION_2	2	X'00000002'
MQZAS_VERSION_3	3	X'00000003'
MQZAS_VERSION_4	4	X'00000004'
MQZAS_VERSION_5	5	X'00000005'
MQZAS_VERSION_6	6	X'00000006'

MQZAT_ * (Tipos de autenticación)

Tabla 441. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAT_INITIAL_CONTEXT	0	X'00000000'
CONTEXTO_CAMBIO_MQZAT_CONTEXTO	1	X'00000001'

MQZCI_ * (Indicador de continuación de servicios instalables)

Tabla 442. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_ * (Estructura de datos de entidad)

Tabla 443. Estructuras de constantes	
Nombre	**Estructura**
MQZED_STRUC_ID	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

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

Tabla 444. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_ * (estructura de parámetros libres)

Tabla 445. Estructuras de constantes	
Nombre	**Estructura**
MQZFP_STRUC_ID	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

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

Tabla 446. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZFP_VERSION_1	1	X'00000001'
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_ * (Estructura de contexto de identidad)

Tabla 447. Estructuras de constantes	
Nombre	**Estructura**
MQZIC_STRUC_ID	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

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

Tabla 448. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZIC_VERSION_1	1	X'00000001'
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_ * (ID de función para servicios)

ID de función comunes a todos los servicios

Tabla 449. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

ID de función para el servicio de autorización

Tabla 450. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT_AUTHORITY	0	X'00000000'

Tabla 450. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPIA_ALL_AUTHORITY	3	X'00000003'
MQZID_DELETE_AUTHORITY	4	X'00000004'
MQZID_SET_AUTHORITY	5	X'00000005'
MQZID_GET_AUTHORITY	6	X'00000006'
MQZID_GET_EXPLICIT_AUTHORITY	7	X'00000007'
MQZID_REFRESH_CACHE	8	X'00000008'
MQZID_ENUMERATE_AUTORITY_DATA	9	X'00000009'
MQZID_AUTOENTICATE_USER	10	X'0000000A'
MQZID_FREE_USER	11	X'0000000B'
MQZID_INQUIRE	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

ID de función para el servicio de nombres

Tabla 451. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT_NAME	0	X'00000000'
MQZID_XX_ENCODE_CASE_ONE nombre_term	1	X'00000001'
MQZID_LOOKUP_NAME	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

ID de función para el servicio de ID de usuario

Tabla 452. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT_USERID	0	X'00000000'
MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_USERID	2	X'00000002'

MQZIO_ * (Opciones de inicialización de servicios instalables)

Tabla 453. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECUNDARIO	1	X'00000001'

MQZNS_ * (Versión de interfaz de servicio de nombres)

Tabla 454. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZNS_VERSION_1	1	X'00000001'

MQZSE_ * (Indicador de enumeración de inicio de servicios instalables)

Tabla 455. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZSE_START	1	X'00000001'
MQZSE_CONTINUE	0	X'00000000'

MQZSL_ * (Indicador de selector de servicios instalables)

Tabla 456. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZSL_NO_DEVUELTO	0	X'00000000'
MQZSL_DEVUELTO	1	X'00000001'

MQZTO_ * (Opciones de terminación de servicios instalables)

Tabla 457. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECUNDARIO	1	X'00000001'

MQZUS_ * (Versión de interfaz de servicio de ID de usuario)

Tabla 458. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZUS_VERSION_1	1	X'00000001'

Tipos de datos utilizados en la MQI

Información sobre los tipos de datos que se pueden utilizar en la interfaz de cola de mensajes (MQI). Descripciones, campos y declaraciones de idioma para los idiomas relevantes con cada tipo de datos.

Tipos de datos y programación para la MQI

Introducción a los tipos de datos Elemental y Estructura, y cómo utilizar la MQI a través de programación C, programación COBOL o programación High Level Assembler .

Tipos de datos elementales

Esta sección contiene información sobre los tipos de datos utilizados en la MQI (o en las funciones de salida). Estos se describen en detalle, seguidos de ejemplos que muestran cómo declarar los tipos de datos elementales en los lenguajes de programación soportados en los temas siguientes.

Los tipos de datos utilizados en la MQI (o en las funciones de salida) son:

- Tipos de datos elementales, o
- Agregados de tipos de datos elementales (matrices o estructuras)

Los siguientes tipos de datos elementales se utilizan en la MQI (o en las funciones de salida):

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales

Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQBOOL	Boolean	El tipo de datos MQBOOL representa un valor booleano. El valor 0 representa false. Cualquier otro valor representa true. Un MQBOOL debe estar alineado como para el tipo de datos MQLONG.
MQBYTE	Byte	El tipo de datos MQBYTE representa un único byte de datos. No se coloca ninguna interpretación particular en el byte; se trata como una serie de bits, y no como un número binario o carácter. No es necesaria ninguna alineación especial. Cuando se envían datos MQBYTE entre gestores de colas que utilizan conjuntos de caracteres o codificaciones diferentes, los datos MQBYTE no se convierten de ninguna manera. Los campos *MsgId* y *CorrelId* de la estructura MQMD son los siguientes. A veces se utiliza una matriz de MQBYTE para representar un área de almacenamiento principal que el gestor de colas no conoce. Por ejemplo, el área puede contener datos de mensaje de aplicación o una estructura. La alineación de los límites de esta zona debe ser compatible con la naturaleza de los datos contenidos en ella. En el lenguaje de programación C, se puede utilizar cualquier tipo de datos para los parámetros de función que se muestran como matrices de MQBYTE. Esto se debe a que estos parámetros siempre se pasan por dirección, y en C el parámetro de función se declara como puntero a vacío.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQBYTEn	Serie de *n* bytes	Cada tipo de datos MQBYTEn representa una serie de *n* bytes, donde *n* puede tomar cualquiera de los valores siguientes: 8, 16, 24, 32, 40 o 128. Cada byte se describe mediante el tipo de datos MQBYTE. No es necesaria ninguna alineación especial. Si los datos de la serie de bytes son más cortos que la longitud definida de la serie, los datos deben rellenarse con nulos para rellenar la serie. Cuando el gestor de colas devuelve series de bytes a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas rellena con nulos la longitud definida de la serie. Las constantes con nombre están disponibles para definir las longitudes de los campos de serie de bytes. Se listan en "Constantes" en la página 61
MQCHAR	Carácter	El tipo de datos MQCHAR representa un carácter de un solo byte o un byte de un carácter de doble byte o de varios bytes. No es necesaria ninguna alineación especial. Cuando se envían datos MQCHAR entre gestores de colas que utilizan diferentes conjuntos de caracteres o codificaciones, los datos MQCHAR normalmente requieren conversión para que los datos se interpreten correctamente. El gestor de colas lo hace automáticamente para los datos MQCHAR en la estructura MQMD. La conversión de datos MQCHAR en los datos del mensaje de aplicación se controla mediante la opción MQGMO_CONVERT especificada en la llamada MQGET; consulte la descripción de esta opción en "MQGMO-Opciones de obtención de mensajes" en la página 367 para obtener más detalles.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQCHARn	Serie de *n* caracteres	Cada tipo de datos MQCHARn representa una serie de *n* caracteres, donde *n* puede tomar cualquiera de los valores siguientes: 4, 8, 12, 20, 28, 32, 48, 64, 128 o 256. Cada carácter se describe mediante el tipo de datos MQCHAR. No es necesaria ninguna alineación especial. Si los datos de la serie son más cortos que la longitud definida de la serie, los datos deben rellenarse con espacios en blanco para rellenar la serie. En algunos casos, se puede utilizar un carácter nulo para finalizar la serie de forma prematura, en lugar de rellenarla con espacios en blanco; el carácter nulo y los caracteres que le siguen se tratan como espacios en blanco, hasta la longitud definida de la serie. Los lugares donde se puede utilizar un nulo se identifican en las descripciones de llamada y tipo de datos. Cuando el gestor de colas devuelve series de caracteres a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con blancos la longitud definida de la serie; el gestor de colas no utiliza el carácter nulo para delimitar la serie. Las constantes con nombre están disponibles que definen las longitudes de los campos de serie de caracteres y se listan en “Constantes” en la página 61.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQFLOAT32	Número de coma flotante de 32 bits	El tipo de datos MQFLOAT32 es un número de coma flotante de 32 bits representado utilizando el formato de coma flotante IEEE estándar. Un MQFLOAT32 debe estar alineado en un límite de 4 bytes. El uso de MQFLOAT32 en C en z/OS requiere el uso del distintivo de compilador FLOAT (IEEE). El uso de MQFLOAT32 en COBOL está limitado a compiladores que dan soporte a números de coma flotante en formato IEEE. Esto puede requerir el uso del distintivo de compilador FLOAT (NATIVE).
MQFLOAT64	Número de coma flotante de 64 bits	El tipo de datos MQFLOAT64 es un número de coma flotante de 64 bits representado utilizando el formato de coma flotante IEEE estándar. Un MQFLOAT64 debe estar alineado en un límite de 8 bytes. El uso de MQFLOAT64 en C en z/OS requiere el uso del distintivo de compilador FLOAT (IEEE). El uso de MQFLOAT64 en COBOL está limitado a compiladores que dan soporte a números de coma flotante en formato IEEE. Esto puede requerir el uso del distintivo de compilador FLOAT (NATIVE).
MQHCONFIG	Descriptor de contexto de configuración	El tipo de datos MQHCONFIG representa un descriptor de contexto de configuración, es decir, el componente que se está configurando para un servicio instalable determinado. Un descriptor de contexto de configuración debe estar alineado en su límite natural. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQHCONN	Descriptor de contexto de conexión	El tipo de datos MQHCONN representa un descriptor de conexión, es decir, la conexión con un gestor de colas determinado. Un descriptor de conexión debe estar alineado en un límite de 4 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.
MQHMSG	Descriptor de contexto de mensaje	El tipo de datos MQHMSG representa un descriptor de mensaje que proporciona acceso a un mensaje. Un descriptor de mensaje debe estar alineado en un límite de 8 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.
MQHOBJ	Descriptor de contexto del objeto	El tipo de datos MQHOBJ representa un descriptor de contexto de objeto que proporciona acceso a un objeto. Un descriptor de contexto de objeto debe estar alineado en un límite de 4 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQINT8	Entero con signo de 8 bits	El tipo de datos MQINT8 es un entero con signo de 8 bits que puede tomar cualquier valor en el rango de -128 a +127, a menos que el contexto restrinja lo contrario.
MQINT16	Entero con signo de 16 bits	El tipo de datos MQINT16 es un entero con signo de 16 bits que puede tomar cualquier valor en el rango de -32 768 a +32 767, a menos que el contexto restrinja lo contrario. Un MQINT16 debe estar alineado en un límite de 2 bytes.
MQINT32	Entero con signo de 32 bits	El tipo de datos MQINT32 es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a + 2 147 483 647, a menos que el contexto restrinja lo contrario. Consulte la definición de MQLONG.
MQINT64	Entero con signo de 64 bits	El tipo de datos MQINT64 es un entero con signo de 64 bits que puede tomar cualquier valor en el rango de -9 223 372 036 854 775 808 a + 9 223 372 036 854 775 807, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a -999 999 999 999 999 999 a +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.
MQLONG	Entero con signo de 32 bits	El tipo de datos MQLONG es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a + 2 147 483 647, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a -999 999 999 a +999 999 999. Un MQLONG debe estar alineado en un límite de 4 bytes.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
IDMQP	Identificador proceso	Identificador de proceso de IBM MQ . Este es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados de FFST™ , pero puede ser diferente del identificador de proceso del sistema operativo.
MQPTR	Puntero	El tipo de datos MQPTR es la dirección de los datos de cualquier tipo. Un puntero debe estar alineado en su límite natural; es un límite de 16 bytes en IBM iy un límite de 8 bytes en otras plataformas. Algunos lenguajes de programación dan soporte a punteros escritos; la MQI también los utiliza en algunos casos (por ejemplo, PMQCHAR y PMQLONG en el lenguaje de programación C).
MQTID	Identificador de hebra	El identificador de hebra de IBM MQ. Es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados de FFST™ , pero puede ser diferente del identificador de hebra del sistema operativo.
MQUINT8	Entero sin signo de 8 bits	El tipo de datos MQUINT8 es un entero sin signo de 8 bits que puede tomar cualquier valor en el rango de 0 a +255, a menos que el contexto restrinja lo contrario.
MQUINT16	Entero sin signo de 16 bits	El tipo de datos MQUINT16 es un entero sin signo de 16 bits que puede tomar cualquier valor en el rango de 0 a +65 535, a menos que el contexto restrinja lo contrario. Un MQUINT16 debe estar alineado en un límite de 2 bytes.
MQUINT32	Entero de 32 bits sin signo	El tipo de datos MQUINT32 es un entero binario sin signo de 32 bits. Consulte la definición de MQULONG.

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQUINT64	Entero de 64 bits sin signo	El tipo de datos MQINT64 es un entero sin signo de 64 bits que puede tomar cualquier valor en el rango de 0 a +18 446 744 073 709 551 615, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a 0 a +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.
MQULONG	Entero de 32 bits sin signo	El tipo de datos MQULONG es un entero binario sin signo de 32 bits que puede tomar cualquier valor en el rango de 0 a + 4 294 967 294, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a 0 a +999 999 999. Un MQULONG debe estar alineado en un límite de 4 bytes.
PMQACH	Puntero	Puntero a una estructura de datos de tipo MQACH
PMQAIR	Puntero	Puntero a una estructura de datos de tipo MQAIR
PMQAXC	Puntero	Puntero a una estructura de datos de tipo MQAXC
PMQAXP	Puntero	Puntero a una estructura de datos de tipo MQAXP
PMQBMHO	Puntero	Puntero a una estructura de datos de tipo MQBMHO
PMQBO	Puntero	Puntero a una estructura de datos de tipo MQBO
PMQBOOL	Puntero	Puntero a datos de tipo MQBOOL
PMQBYTE	Puntero	Puntero a datos de tipo MQBYTE
PMQBYTEn	Puntero	Puntero a datos de tipo MQBYTEn, donde n puede ser 8, 16, 24, 32, 40, 128
PMQCBC	Puntero	Puntero a una estructura de datos de tipo MQCBC
PMQCBD	Puntero	Puntero a una estructura de datos de tipo MQCBD
PMQCHAR	Puntero	Puntero a datos de tipo MQCHAR

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
PMQCHARN	Puntero	Puntero a un tipo de datos de MQCHARN, donde n puede ser 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCHARV	Puntero	Puntero a una estructura de datos de tipo MQCHARV
PMQCIH	Puntero	Puntero a una estructura de datos de tipo MQCIH
PMQCMHO	Puntero	Puntero a una estructura de datos de tipo MQCMHO
PMQCNO	Puntero	Puntero a una estructura de datos de tipo MQCNO
PMQCSP	Puntero	Puntero a una estructura de datos de tipo MQCSP
PMQCTLO	Puntero	Puntero a una estructura de datos de tipo MQCTLO
PMQDH	Puntero	Puntero a una estructura de datos de tipo MQDH
PMQDHO	Puntero	Puntero a una estructura de datos de tipo MQDHO
PMQDLH	Puntero	Puntero a una estructura de datos de tipo MQDLH
PMQDMHO	Puntero	Puntero a una estructura de datos de tipo MQDMHO
PMQDMPO	Puntero	Puntero a una estructura de datos de tipo MQDMPO
PMQEPH	Puntero	Puntero a una estructura de datos de tipo MQEPH
PMQFLOAT32	Puntero	Puntero a una estructura de datos de tipo MQFLOAT32
PMQFLOAT64	Puntero	Puntero a una estructura de datos de tipo MQFLOAT64
PMQFUNC	Puntero	Puntero a una función
PMQGMO	Puntero	Puntero a una estructura de datos de tipo MQGMO
PMQHCONFIG	Puntero	Puntero a datos de tipo MQHCONFIG
PMQHCONN	Puntero	Puntero a datos de tipo MQHCONN
PMQHMSG	Puntero	Puntero a datos de tipo MQHMSG
PMQHOBJ	Puntero	Puntero a datos de tipo MQHOBJ

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
PMQIIH	Puntero	Puntero a una estructura de datos de tipo MQIIH
PMQIMPO	Puntero	Puntero a una estructura de datos de tipo MQIMPO
PMQINT8	Puntero	Puntero a datos de tipo MQINT8
PMQINT16	Puntero	Puntero a datos de tipo MQINT16
PMQINT32	Puntero	Puntero a datos de tipo MQINT32
PMQINT64	Puntero	Puntero a datos de tipo MQINT64
PMQLONG	Puntero	Puntero a datos de tipo MQLONG
PMQMD	Puntero	Puntero a estructura de tipo MQMD
PMQMDE	Puntero	Puntero a una estructura de datos de tipo MQMDE
PMQMD1	Puntero	Puntero a una estructura de datos de tipo MQMD1
PMQMD2	Puntero	Puntero a una estructura de datos de tipo MQMD2
PMQMHBO	Puntero	Puntero a una estructura de datos de tipo MQMHBO
PMQOD	Puntero	Puntero a una estructura de datos de tipo MQOD
PMQOR	Puntero	Puntero a una estructura de datos de tipo MQOR
PMQPD	Puntero	Puntero a una estructura de datos de tipo MQPD
PMQPID	Puntero	Puntero a un identificador de proceso
PMQMD	Puntero	Puntero a una estructura de datos de tipo MQMD
PMQPMO	Puntero	Puntero a una estructura de datos de tipo MQPMO
PMQPTR	Puntero	Puntero a datos de tipo MQPTR
PMQRFH	Puntero	Puntero a una estructura de datos de tipo MQRFH
PMQRFH2	Puntero	Puntero a una estructura de datos de tipo MQRFH2
PMQRMH	Puntero	Puntero a una estructura de datos de tipo MQRMH
PMQRR	Puntero	Puntero a una estructura de datos de tipo MQRR

Tabla 459. Nombres, tipos y descripciones de tipos de datos elementales (continuación)		
Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
PMQSCO	Puntero	Puntero a una estructura de datos de tipo MQSCO
PMQD	Puntero	Puntero a una estructura de datos de tipo MQSD
PMQSMPO	Puntero	Puntero a una estructura de datos de tipo MQSMPO
PMQSRO	Puntero	Puntero a una estructura de datos de tipo MQSRO
PSSTS	Puntero	Puntero a una estructura de datos de tipo MQSTS
PMQTID	Puntero	Puntero a un ID de hebra
PMQTM	Puntero	Puntero a una estructura de datos de tipo MQTM
PMQTMC2	Puntero	Puntero a una estructura de datos de tipo MQTMC2
PMQUINT8	Puntero	Puntero a un tipo de datos de MQUINT8
PMQUINT16	Puntero	Puntero a un tipo de datos de MQUINT16
PMQUINT32	Puntero	Puntero a un tipo de datos de MQUINT32
PMQUINT64	Puntero	Puntero a un tipo de datos de MQUINT64
PMQULONG	Puntero	Puntero a un tipo de datos de MQULONG
PMQVOID	Puntero	
PMQWIH	Puntero	Puntero a una estructura de datos de tipo MQWIH
PMQXQH	Puntero	Puntero a una estructura de datos de tipo MQXQH

Declaraciones C

Tabla 460. Nombres y representaciones de tipos de datos C	
Tipo de datos	**Representación**
MQBOOL	`typedef MQLONG MQBOOL;`
MQBYTE	`typedef unsigned char MQBYTE;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
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];`
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];`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
MQFLOAT32	`typedef float MQFLOAT32;`
MQFLOAT64	`typedef double MQFLOAT64;`
MQHCONFIG	`typedef void MQPOINTER MQHCONFIG;`
MQHCONN	`typedef MQLONG MQHCONN;`
MQHOBJ	`typedef MQLONG MQHOBJ;`
MQINT8	`typedef signed char MQINT8;`
MQINT16	`typedef short MQINT16;`

MQINT64	UNIX En UNIXde 64 bits:
	`typedef long;`
	UNIX En AIXde 32 bits, Solaris:
	`typedef int64_t;`
	z/OS Linux IBM i En Linux, IBM iy z/OS:
	`typedef long long;`
	Windows En Windows:
	`typedef _int64;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**

MQLONG	IBM i En IBM i:
	`typedef long MQLONG;`
	ULW z/OS En otras plataformas:
	`if defined(MQ_64_BIT)`
	`  typedef int MQLONG;`
	`else`
	`  typedef long MQLONG;`

IDMQP	`typedef MQLONG MQPID;`
MQPTR	`typedef void MQPOINTER MQPTR;`
MQTID	`typedef MQLONG MQTID;`
MQUINT8	`typedef unsigned char MQUINT8;`
MQUINT16	`typedef unsigned short MQUINT16;`

MQUINT64	UNIX En UNIXde 64 bits:
	`typedef unsigned long;`
	UNIX En AIXde 32 bits, Solaris:
	`typedef uint64_t;`
	z/OS Linux IBM i En Linux, IBM iy z/OS:
	`typedef unsigned long long;`
	Windows En Windows:
	`typedef unsigned _int64;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**

MQULONG	IBM i En IBM i:
	`typedef unsigned long MQULONG;`
	ULW z/OS En otras plataformas:
	`if defined(MQ_64_BIT)`
	`  typedef unsigned int MQULONG;`
	`else`
	`  typedef unsigned long MQULONG;`

PMQBO	`typedef MQBO MQPOINTER PMQBO;`
PMQBOOL	`typedef MQBOOL MQPOINTER PMQBOOL;`
PMQBYTE	`typedef MQBYTE MQPOINTER PMQBYTE;`
PMQBYTE8	`typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];`
PMQBYTE16	`typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];`
PMQBYTE24	`typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];`
PMQBYTE32	`typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];`
PMQBYTE40	`typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];`
PMQBYTE128	`typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];`
PMQCHAR	`typedef MQCHAR MQPOINTER PMQCHAR;`
PMQCHAR4	`typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];`
PMQCHAR8	`typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
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;`
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;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
PMQHCONFIG	`typedef MQHCONFIG MQPOINTER PMQHCONFIG;`
PMQHCONN	`typedef MQHCONN MQPOINTER PMQHCONN;`
PMQHOBJ	`typedef MQHOBJ MQPOINTER PMQHOBJ;`
PMQIIH	`typedef MQIIH MQPOINTER PMQIIH;`
PMQINT8	`typedef MQINT8 MQPOINTER PMQINT8;`
PMQINT16	`typedef MQINT16 MQPOINTER PMQINT16;`
PMQLONG	`typedef MQLONG MQPOINTER PMQLONG;`
PMQMD	`typedef MQMD MQPOINTER PMQMD;`
PMQMD1	`typedef MQMD1[1] MQPOINTER PMQMD1[1];`
PMQMDE	`typedef MQMDE MQPOINTER PMQMDE;`
PMQOD	`typedef MQOD MQPOINTER PMQOD;`
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;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
PMQTM	`typedef MQTM MQPOINTER PMQTM;`
PMQTMC2	`typedef MQTMC2[2] MQPOINTER PMQTMC2[2];`
PMQUINT8	`typedef MQUINT8 MQPOINTER PMQUINT8;`
PMQUINT16	`typedef MQUINT16 MQPOINTER PMQUINT16;`
PMQULONG	`typedef MQULONG MQPOINTER PMQULONG;`
PMQVOID	`typedef void MQPOINTER PMQVOID;`
PMQWIH	`typedef MQWIH MQPOINTER PMQWIH;`
PMQXQH	`typedef MQXQH MQPOINTER PMQXQH;`
PMQBO	`typedef PMQBO MQPOINTER PPMQBO;`
PMQBYTE	`typedef PMQBYTE MQPOINTER PPMQBYTE;`
PMQCHAR	`typedef PMQCHAR MQPOINTER PPMQCHAR;`
PMQCNO	`typedef PMQCNO MQPOINTER PPMQCNO;`
PMQGMO	`typedef PMQGMO MQPOINTER PPMQGMO;`
PMQHCONN	`typedef PMQHCONN MQPOINTER PPMQHCONN;`
PMQHOBJ	`typedef PMQHOBJ MQPOINTER PPMQHOBJ;`
PMQLONG	`typedef PMQLONG MQPOINTER PPMQLONG;`

Tabla 460. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	**Representación**
PPMQMD	`typedef PMQMD MQPOINTER PPMQMD;`
PMQOD	`typedef PMQOD MQPOINTER PPMQOD;`
PMQPMO	`typedef PMQPMO MQPOINTER PPMQPMO;`
PMQULONG	`typedef PMQULONG MQPOINTER PPMQULONG;`
PMQVOID	`typedef PMQVOID MQPOINTER PPMQVOID;`
Donde `defined(MQ_64_BIT)` significa una plataforma de 64 bits.	

Consulte "Tipos de datos" en la página 265 para obtener una descripción de la variable de macro MQPOINTER.

Declaraciones COBOL

Tabla 461. Nombres y representaciones de tipos de datos COBOL	
Tipo de datos	**Representación**
MQBOOL	`PIC S9(9) BINARY`
MQBYTE	`PIC X`
MQBYTE8	`PIC X(8)`
MQBYTE16	`PIC X(16)`
MQBYTE24	`PIC X(24)`
MQBYTE32	`PIC X(32)`
MQBYTE40	`PIC X(40)`
MQCHAR	`PIC X`

Tabla 461. Nombres y representaciones de tipos de datos COBOL (continuación)	
Tipo de datos	**Representación**
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)`
MQFLOAT32	`USAGE COMP-1`
MQFLOAT64	`USAGE COMP-2`

MQHCONN	En z/OS
	`PIC S9(9) COMP-5`
	En otras plataformas
	`PIC S9(9) BINARY`

MQHOBJ	`PIC S9(9) BINARY`

Tabla 461. Nombres y representaciones de tipos de datos COBOL (continuación)	
Tipo de datos	**Representación**
MQINT8	`PIC S9(2) BINARY`
MQINT16	`PIC S9(4) BINARY`
MQINT64	`PIC S9(18) BINARY`
MQLONG	`PIC S9(9) BINARY`
MQPTR	`POINTER`
MQUINT8	`PIC 9(2) BINARY`
MQUINT16	`PIC 9(4) BINARY`
MQUINT64	`PIC 9(18) BINARY`
MQULONG	`PIC 9(9) BINARY`

Declaraciones PL/I
PL/I está soportado en z/OS.

Tabla 462. Nombres y representaciones de tipos de datos PL/I	
Tipo de datos	**Representación**
MQBOOL	`fixed bin(31)`
MQBYTE	`char(1)`
MQBYTE8	`char(8)`
MQBYTE16	`char(16)`
MQBYTE24	`char(24)`

Tabla 462. Nombres y representaciones de tipos de datos PL/I (continuación)	
Tipo de datos	**Representación**
MQBYTE32	`char(32)`
MQBYTE40	`char(40)`
MQCHAR	`char(1)`
MQCHAR4	`char(4)`
MQCHAR8	`char(8)`
MQCHAR12	`char(12)`
MQCHAR20	`char(20)`
MQCHAR28	`char(28)`
MQCHAR32	`char(32)`
MQCHAR48	`char(48)`
MQCHAR64	`char(64)`
MQCHAR128	`char(128)`
MQCHAR256	`char(256)`
MQFLOAT32	`binary float(21) ieee`
MQFLOAT64	`binary float(52) ieee`
MQHCONN	`fixed bin(31)`

Tabla 462. Nombres y representaciones de tipos de datos PL/I (continuación)	
Tipo de datos	**Representación**
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)`

Declaraciones de ensamblador System/390
El ensamblador System/390 sólo está soportado en z/OS .

Tabla 463. Nombres y representaciones de tipos de datos de ensamblador System/390	
Tipo de datos	**Representación**
MQBOOL	`DS  F`
MQBYTE	`DS  XL1`
MQBYTE8	`DS  XL8`
MQBYTE16	`DS  XL16`

Tabla 463. Nombres y representaciones de tipos de datos de ensamblador System/390 (continuación)	
Tipo de datos	**Representación**
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`
MQCHAR128	`DS CL128`
MQCHAR256	`DS CL256`
MQFLOAT32	`DS EB`
MQFLOAT64	`DS DB`

Tabla 463. Nombres y representaciones de tipos de datos de ensamblador System/390 (continuación)	
Tipo de datos	**Representación**
MQHCONN	`DS F`
MQHOBJ	`DS F`
MQINT8	`DS XL1`
MQINT16	`DS H`
MQINT64	`DS D`
MQLONG	`DS F`
MQPTR	`DS F`
MQUINT8	`DS XL1`
MQUINT16	`DS H`
MQUINT64	`DS D`
MQULONG	`DS F`

Tipos de datos de estructura

Resumen de los tipos de datos de estructura, reglas para correlacionar las estructuras MQI de forma coherente y convenios utilizados en cada descripción de tipo de datos de estructura.

- "Resumen de los tipos de datos de estructura utilizados en llamadas MQI o funciones de salida" en la página 261
- "Resumen de los tipos de datos de estructura utilizados en los datos de mensaje" en la página 262
- "Reglas para correlacionar las estructuras MQI de forma coherente" en la página 262
- "Convenciones utilizadas en cada descripción de tipo de datos de estructura" en la página 263

Resumen de los tipos de datos de estructura utilizados en llamadas MQI o funciones de salida

Tabla 464. Tipos de datos de estructura utilizados en llamadas MQI o funciones de salida

Estructura	Descripción	Llamadas donde se utilizan
MQACH	Cabecera de cadena de salida de API	
MQAIR	Registro de información de autenticación	MQCONNX
MQAXC	Contexto de salida de API	
MQAXP	Parámetro de salida de API	
MQBMHO	Opciones de almacenamiento intermedio a manejador de mensajes	MQBUFMH
MQBO	Opciones de inicio	MQBEGIN
MQCBD	Descriptor de devolución de llamada	MQCB
MQCBO	Opciones de creación de paquete	mqCreateBag
MQCHARV	Serie de longitud variable	MQINQMP
MQCNO	Opciones de conexión	MQCONNX
MQCSP	Parámetros de seguridad	MQCONNX
MQCTLO	Opciones de devolución de llamada	MQCTL
MQDMPO	Suprimir opciones de propiedad de mensaje	MQDLTMP
MQGMO	Opciones para obtener mensaje	MQGet
MQIMPO	Consultar opciones de propiedad de mensaje	MQINQMP
MQMD	Descriptor de mensaje	MQBUFMH, MQMHBUF, MQCB, MQGET, MQPUT, MQPUT1
MQMHBO	Manejador de mensajes para opciones de almacenamiento intermedio	MQMHBUF
MQOD	Descriptor de objetos	MQOPEN, MQPUT1
MQOR	Registro de objeto	MQOPEN, MQPUT1
MQPD	Descriptor de propiedad	MQSETMP
MQPMO	Opciones para transferir mensaje	MQPUT, MQPUT1
MQPMR	Registro de transferencia de mensaje	MQPUT, MQPUT1
MQRR	Registro de respuestas	MQOPEN, MQPUT, MQPUT1
MQSCO	Opciones de configuración de TLS	MQCONNX
MQSD	Descriptor de suscripción	MQSUB

Tabla 464. Tipos de datos de estructura utilizados en llamadas MQI o funciones de salida (continuación)		
Estructura	**Descripción**	**Llamadas donde se utilizan**
MQSMPO	Establecer opción de propiedad de mensaje	MQSETMP
MQSRO	Opciones de solicitud de suscripción	MQSUBRQ
MQSTS	Estructura de informes de estado	MQSTAT

Resumen de los tipos de datos de estructura utilizados en los datos de mensaje

Tabla 465. Tipos de datos de estructura utilizados en datos de mensaje	
Estructura	**Descripción**
MQCIH	Cabecera de información de CICS
MQCFH	Cabecera PCF
MQEPH	Cabecera PCF incluida
MQDH	Cabecera de distribución
MQDLH	Cabecera de mensaje no entregado (mensaje no entregado)
MQIIH	Cabecera de información de IMS
MQMDE	Extensión del descriptor de mensaje
MQRFH	Cabecera de reglas y formato
MQRFH2	Cabecera de reglas y formato 2
MQRMH	Cabecera de mensaje de referencia
MQTM	Mensaje de desencadenante
MQTMC2	Mensaje desencadenante (formato de caracteres 2)
MQWIH	Cabecera de información de trabajo
MQXQH	Cabecera de cola de transmisión

Nota: La estructura MQDXP (parámetro de salida de conversión de datos) se describe en “salida de conversión de datos” en la página 923, junto con las llamadas de conversión de datos asociadas.

Reglas para correlacionar las estructuras MQI de forma coherente

Los lenguajes de programación varían en su nivel de soporte para estructuras, y se adoptan determinadas reglas y convenios para correlacionar las estructuras MQI de forma coherente en cada lenguaje de programación:

1. Las estructuras deben estar alineadas en sus límites naturales.
   - La mayoría de las estructuras MQI requieren una alineación de 4 bytes.
   - En IBM i, las estructuras que contienen punteros requieren una alineación de 16 bytes; estos son: MQCNO, MQOD, MQPMO.
2. Cada campo de una estructura debe estar alineado en su límite natural.
   - Los campos con tipos de datos que equivalen a MQLONG deben estar alineados en límites de 4 bytes.

- Los campos con tipos de datos que equivalen a MQPTR deben estar alineados en límites de 16 bytes en IBM iy en límites de 4 bytes en otros entornos.
- Otros campos están alineados en límites de 1 byte.

3. La longitud de una estructura debe ser un múltiplo de su alineación de límite.
   - La mayoría de las estructuras MQI tienen longitudes que son múltiplos de 4 bytes.
   - En IBM i, las estructuras que contienen punteros tienen longitudes que son múltiplos de 16 bytes.
4. Cuando sea necesario, se deben añadir campos o bytes de relleno para garantizar el cumplimiento de las reglas anteriores.

Convenciones utilizadas en cada descripción de tipo de datos de estructura

La descripción de cada tipo de datos de estructura incluye:

- Una visión general de la finalidad y el uso de la estructura
- Descripciones de los campos de la estructura, en un formato independiente del lenguaje de programación
- Ejemplos de cómo se declara la estructura en cada uno de los lenguajes de programación soportados

La descripción de cada tipo de datos de estructura contiene las secciones siguientes:

Nombre de estructura
El nombre de la estructura, seguido de un resumen de los campos de la estructura.

Visión general
Una breve descripción del propósito y uso de la estructura.

Campos
Descripciones de los campos. Para cada campo, el nombre del campo va seguido de su tipo de datos elemental entre paréntesis (). En el texto, los nombres de campo se muestran utilizando un tipo de letra cursiva; por ejemplo, *Version*.

También hay una descripción de la finalidad del campo, junto con una lista de los valores que el campo puede tomar. Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQGMO_STRUC_ID. Un conjunto de constantes que tienen el mismo prefijo se muestra utilizando el carácter *, por ejemplo: MQIA_ *.

En las descripciones de los campos, se utilizan los términos siguientes:

entrada
Proporcione información en el campo cuando realice una llamada.

salida
El gestor de colas devuelve información en el campo cuando la llamada se completa o falla.

entrada/salida
Proporcione información en el campo cuando realice una llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Valores iniciales
Una tabla que muestra los valores iniciales para cada campo en los archivos de definición de datos proporcionados con la MQI.

Declaración C
Declaración típica de la estructura en C.

declaración COBOL
Declaración típica de la estructura en COBOL.

declaración PL/I
Declaración típica de la estructura en PL/I.

Declaración High Level Assembler
Declaración típica de la estructura en lenguaje ensamblador System/390 .

Declaración de Visual Basic
Declaración típica de la estructura en Visual Basic.

Programación C

Información para ayudarle a utilizar la MQI del lenguaje de programación C.

- “Archivos de cabecera” en la página 264
- “Funciones” en la página 264
- “Parámetros con tipo de datos no definido” en la página 265
- “Tipos de datos” en la página 265
- “Manipulación de series binarias” en la página 265
- “Manipulación de series de caracteres” en la página 265
- “Valores iniciales para estructuras” en la página 266
- “Valores iniciales para estructuras dinámicas” en la página 266
- “Utilizar desde C++” en la página 267
- “Convenios de notación” en la página 267

Archivos de cabecera

Tabla 466. Archivos de cabecera C

Archivo	Contenido
CMQC	Prototipos de función, tipos de datos y constantes con nombre para la MQI principal
CMQXC	Prototipos de función, tipos de datos y constantes con nombre para la salida de conversión de datos
CMQEC	Prototipos de función, tipos de datos y constantes con nombre para la MQI principal, la salida de conversión de datos y la estructura de puntos de entrada de interfaz (CMQEC incluye CMQXC y CMQC.)
CMQSTRC	Funciones que convierten definiciones de constantes MQI en el equivalente de texto. **Atención:** z/OS V 9.1.0 Aplicable a z/OS desde IBM MQ 9.1. Los programas que utilizan este archivo de cabecera deben compilarse con la opción de compilador LONGNAME.

Para mejorar la portabilidad de aplicaciones, codifique el nombre del archivo de cabecera en minúsculas en la directiva de preprocesador `#include`:

```
#include "cmqec.h"
```

Funciones

No es necesario especificar todos los parámetros que se pasan por dirección cada vez que se invoca una función.

- Pasar parámetros que son *de sólo entrada* y de tipo MQHCONN, MQHOBJ o MQLONG por valor.
- Pase todos los demás parámetros por dirección.

Cuando no sea necesario un parámetro determinado, utilice un puntero nulo como parámetro en la invocación de la función, en lugar de la dirección de los datos del parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la función; en terminología C, esto significa que todas las funciones devuelven `void`.

Los atributos de la función se definen mediante la variable de macro MQENTRY; el valor de esta variable de macro depende del entorno.

Parámetros con tipo de datos no definido

El parámetro **Buffer** en las funciones MQGET, MQPUT y MQPUT1 tiene un tipo de datos no definido. Este parámetro se utiliza para enviar y recibir los datos de mensaje de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero normalmente es más conveniente declararlos como la estructura particular que describe el diseño de los datos en el mensaje. Declare el parámetro de función real como puntero a vacío y especifique la dirección de cualquier tipo de datos como parámetro en la invocación de la función.

Tipos de datos

Defina todos los tipos de datos utilizando la sentencia `typedef` de C. Para cada tipo de datos, defina también el tipo de datos de puntero correspondiente. El nombre del tipo de datos de puntero es el nombre del tipo de datos elemental o de estructura prefijado con la letra P para denotar un puntero. Defina los atributos del puntero utilizando la variable de macro MQPOINTER; el valor de esta variable de macro depende del entorno. A continuación se muestra cómo declarar tipos de datos de puntero:

```
#define MQPOINTER *                /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;    /* pointer to MQMD */
```

Manipulación de series binarias

Declare series de datos binarios como uno de los tipos de datos MQBYTEn.

Siempre que copie, compare o establezca campos de este tipo, utilice las funciones C **memcpy**, **memcmp**o **memset** ; por ejemplo:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls    */
       MQMI_NONE,                 /* ...using named constant       */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
       0x00,                      /* ...using a different method   */
       sizeof(MQBYTE24));
```

No utilice las funciones de serie **strcpy**, **strcmp**, **strncpy**o **strncmp**, porque no funcionan correctamente para los datos declarados con los tipos de datos MQBYTEn.

Manipulación de series de caracteres

Cuando el gestor de colas devuelve los datos de caracteres a la aplicación, el gestor de colas siempre rellena los datos de caracteres con espacios en blanco hasta la longitud definida del campo. El gestor de colas *no* devuelve series terminadas en nulo.

Por lo tanto, al copiar, comparar o concatenar dichas series, utilice las funciones de serie **strncpy**, **strncmp**o **strncat**.

No utilice las funciones de serie que requieren que la serie termine con un valor nulo (**strcpy**, **strcmp**, **strcat**). Además, no utilice la función **strlen** para determinar la longitud de la serie; utilice en su lugar la función **sizeof** para determinar la longitud del campo.

Valores iniciales para estructuras

Los archivos de cabecera definen diversas variables de macro que puede utilizar para proporcionar valores iniciales para las estructuras de MQ al declarar instancias de dichas estructuras.

Estas variables de macro tienen nombre con el formato MQxxx_DEFAULT, donde MQxxx representa el nombre de la estructura. Se utilizan de la siguiente manera:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Para algunos campos de caracteres (por ejemplo, los campos *StrucId* que aparecen en la mayoría de las estructuras, o el campo *Format* que aparece en MQMD), la MQI define valores concretos que son válidos. Para cada uno de los valores válidos, se proporcionan *dos* variables de macro:

- Una variable de macro define el valor como una serie con una longitud, excluyendo las coincidencias nulas implícitas, exactamente la longitud definida del campo. Por ejemplo, para el campo *Format* en MQMD se proporciona la siguiente variable de macro (¬ representa un único carácter en blanco):

  ```
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Utilice este formulario con las funciones `memcpy` y `memcmp` .
- La otra variable de macro define el valor como una matriz de caracteres; el nombre de esta variable de macro es el nombre del formato de serie con el sufijo _ARRAY. Por ejemplo:

  ```
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Utilice este formulario para inicializar el campo cuando declare una instancia de la estructura con valores distintos de los proporcionados por la variable de macro MQMD_DEFAULT. (Esto no siempre es necesario; en algunos entornos puede utilizar el formato de serie del valor en ambas situaciones. Sin embargo, puede utilizar el formato de matriz para las declaraciones, porque esto es necesario para la compatibilidad con el lenguaje de programación C++.)

Valores iniciales para estructuras dinámicas

Cuando se necesita un número variable de instancias de una estructura, las instancias se crean normalmente en el almacenamiento principal obtenido dinámicamente utilizando las funciones `calloc` o `malloc` . Para inicializar los campos en dichas estructuras, tenga en cuenta la técnica siguiente:

1. Declare una instancia de la estructura utilizando la variable de macro MQxxx_DEFAULT adecuada para inicializar la estructura. Esta instancia se convierte en el modelo para otras instancias:

   ```
   MQMD Model = {MQMD_DEFAULT}; /* declare model instance */
   ```

   Las palabras clave `static` o `auto` se pueden codificar en la declaración para proporcionar la duración estática o dinámica de la instancia de modelo, según sea necesario.
2. Utilice las funciones `calloc` o `malloc` para obtener almacenamiento para una instancia dinámica de la estructura:

   ```
   PMQMD Instance;
   Instance = malloc(sizeof(MQMD)); /* get storage for dynamic instance */
   ```

3. Utilice la función `memcpy` para copiar la instancia de modelo en la instancia dinámica:

```
memcpy(Instance,&Model,sizeof(MQMD)); /* initialize dynamic instance */
```

Utilizar desde C++

Para el lenguaje de programación C++, los archivos de cabecera contienen las siguientes sentencias adicionales que sólo se incluyen cuando se utiliza un compilador C++:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Convenios de notación

Esta información muestra cómo invocar las funciones y declarar parámetros.

En algunos casos, los parámetros son matrices con un tamaño que no es fijo. Para estos, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

Programación COBOL

Esta sección contiene información para ayudarle a utilizar la MQI del lenguaje de programación COBOL.

Programación de High Level Assembler

Información para ayudarle a utilizar la MQI del lenguaje de programación System/390 Assembler.

- “Macros” en la página 267
- “Estructuras” en la página 268
- “Macro MQVERA” en la página 268
- “Convenios de notación” en la página 268

Macros

Hay dos macros para constantes con nombre y una macro para cada una de las estructuras. Estos archivos se resumen en la tabla siguiente.

Tabla 467. Macros de Assembler

Archivo	Contenido
CMQA	Constantes con nombre (equates) para MQI principal
CMQCIHA	Estructura de cabecera de información de CICS
CMQCNOA	Estructura de opciones de conexión
CMQDLHA	Estructura de cabecera de mensaje no entregado
CMQDXPA	Estructura de parámetros de salida de conversión de datos
CMQGMOA	Estructura de opciones de obtención de mensajes
CMQIIHA	Estructura de cabecera de información de IMS
CMQMDA	Estructura del descriptor de mensaje

Tabla 467. Macros de Assembler (continuación)	
Archivo	**Contenido**
CMQMDEA	Estructura de extensión de descriptor de mensaje
CMQODA	Estructura de descriptor de objeto
CMQPMOA	Estructura de opciones de colocación de mensaje
CMQRFHA	Reglas y estructura de cabecera de formato
CMQRFH2A	Reglas y estructura de cabecera de formato versión 2
CMQRMHA	Estructura de cabecera de mensaje de referencia
CMQTMA	Estructura de mensajes desencadenantes
CMQTMC2A	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQVERA	Control de versión de estructura
CMQWIHA	Estructura de cabecera de información de trabajo
CMQXA	Constantes con nombre para la salida de conversión de datos
CMQXPA	Estructura de parámetros de salida cruzada de API
CMQXQHA	Estructura de cabecera de cola de transmisión

Estructuras

Las estructuras son generadas por macros que tienen varios parámetros para controlar la acción de la macro. Consulte "Estructuras" en la página 268

Macro MQVERA

Esta macro le permite establecer el valor predeterminado que se utilizará para el parámetro `DCLVER` en las macros de estructura.

El valor especificado por CMQVERA lo utiliza la macro de estructura sólo si omite el parámetro `DCLVER` de la invocación de la macro de estructura. El valor predeterminado se establece codificando la macro CMQVERA con el parámetro `DCLVER` :

DCLVER=ACTUAL
: La versión predeterminada se establece en la versión actual (más reciente).

DCLVER=ESPECIFICADO
: La versión predeterminada se establece en la versión especificada por el parámetro `VERSION` .

Debe especificar el parámetro **DCLVER** y el valor debe estar en mayúsculas. El valor establecido por CMQVERA sigue siendo el valor predeterminado hasta la siguiente invocación de CMQVERA o el final del ensamblaje. Si omite CMQVERA, el valor predeterminado es `DCLVER=CURRENT`.

Convenios de notación

Otros temas muestran cómo invocar las llamadas y declarar parámetros. En algunos casos, los parámetros son matrices o series de caracteres con un tamaño que no está fijado para el cual, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

Estructuras
Las estructuras son generadas por macros que tienen varios parámetros para controlar la acción de la macro.

Nota: De vez en cuando se introducen nuevas versiones de las estructuras IBM MQ . Los campos adicionales en una nueva versión pueden hacer que una estructura que anteriormente era menor que 256 bytes sea mayor que 256 bytes. Debido a esto, escriba las instrucciones del ensamblador que están pensadas para copiar una estructura IBM MQ , o para establecer una estructura IBM MQ en nulos, para trabajar correctamente con estructuras que pueden tener más de 256 bytes. De forma alternativa, utilice el parámetro de macro DCLVER o la macro CMQVERA con el parámetro VERSION para declarar una versión específica de la estructura.

- "Especificación del nombre de la estructura" en la página 269
- "Especificación del formato de la estructura" en la página 269
- "Control de la versión de la estructura" en la página 269
- "Declaración de una estructura incluida en otra" en la página 270
- "Especificación del valor inicial de un campo" en la página 270
- "Control del listado" en la página 270

Especificación del nombre de la estructura

Para declarar más de una instancia de una estructura, la macro prefija el nombre de cada campo de la estructura con una serie especificada por el usuario y un carácter de subrayado.

La serie utilizada es la etiqueta especificada en la invocación de la macro. Si no se especifica ninguna etiqueta, se utiliza el nombre de la estructura para construir el prefijo:

```
* Declare two object descriptors
         CMQODA  ,        Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA  ,        Prefix used="MY_MQOD_"
```

Las declaraciones de estructura que se muestran en esta sección utilizan el prefijo predeterminado.

Especificación del formato de la estructura

La macro puede generar declaraciones de estructura en uno de dos formatos, controlados por el parámetro DSECT :

DSECT=YES
: Se utiliza una instrucción DSECT de ensamblador para iniciar una nueva sección de datos; la definición de estructura sigue inmediatamente a la sentencia DSECT . La etiqueta en la invocación de macro se usa como nombre de la sección de datos; si no se especifica ninguna etiqueta, se usará el nombre de la estructura.

DSECT=NO
: Las instrucciones DC del ensamblador se utilizan para definir la estructura en la posición actual de la rutina. Los campos se inicializan con valores, que se pueden especificar codificando los parámetros relevantes en la invocación de macro. Los campos para los que no se especifican valores en la invocación de la macro se inicializan con valores predterminados.

El valor especificado debe estar en mayúsculas. Si no se especifica el parámetro DSECT , se presupone DSECT = NO .

Control de la versión de la estructura

De forma predeterminada, las macros siempre declaran la versión más reciente de cada estructura.

Aunque puede utilizar el parámetro de macro VERSION para especificar un valor para el campo *Version* en la estructura, ese parámetro define el valor inicial para el campo *Version* y no controla la versión de la estructura declarada realmente. Para controlar la versión de la estructura que se declara, utilice el parámetro DCLVER :

DCLVER=ACTUAL
: La versión declarada es la versión actual (más reciente).

DCLVER=ESPECIFICADO
La versión declarada es la versión especificada por el parámetro VERSION . Si omite el parámetro VERSION , el valor predeterminado es la versión 1.

Si especifica el parámetro VERSION , el valor debe ser una constante numérica autodefinida o la constante con nombre para la versión necesaria (por ejemplo, MQCNO_VERSION_3). Si especifica algún otro valor, la estructura se declara como si se hubiera especificado DCLVER=CURRENT , incluso si el valor de VERSION se resuelve en un valor válido.

El valor especificado debe estar en mayúsculas. Si omite el parámetro DCLVER , el valor utilizado se toma de la variable de macro global MQDCLVER . Puede establecer esta variable utilizando la macro CMQVERA.

Declaración de una estructura incluida en otra

Para declarar una estructura como componente de otra estructura, utilice el parámetro ANIDADO :

NESTED=SÍ
La declaración de estructura está anidada dentro de otra.

NESTED=NO
La declaración de estructura no está anidada dentro de otra.

El valor especificado debe estar en mayúsculas. Si omite el parámetro ANIDADO , se presupone NESTED=NO .

Especificación del valor inicial de un campo

Especifique el valor que se utilizará para inicializar un campo en una estructura codificando el nombre de ese campo (sin el prefijo) como parámetro en la invocación de la macro, acompañado del valor necesario.

Por ejemplo, para declarar una estructura de descriptor de mensaje con el campo *MsgType* inicializado con MQMT_REQUEST y el campo *ReplyToQ* inicializado con la serie "MY_REPLY_TO_QUEUE", utilice lo siguiente:

```
MY_MQMD  CMQMDA  MSGTYPE=MQMT_REQUEST,                              X
               REPLYTOQ=MY_REPLY_TO_QUEUE
```

Si especifica una constante con nombre (equate) como un valor en la invocación de la macro, utilice la macro CMQA para definir la constante con nombre. No encierre los valores de serie de caracteres entre comillas simples.

Control del listado

Controle el aspecto de la declaración de estructura en el listado del ensamblador utilizando el parámetro LIST :

LIST=YES
La declaración de estructura aparece en el listado del ensamblador.

LIST=NO
La declaración de estructura no aparece en el listado del ensamblador.

El valor especificado debe estar en mayúsculas. Si omite el parámetro LIST , se presupone LIST = NO .

MQAIR-Registro de información de autenticación

La estructura MQAIR permite a una aplicación que se ejecuta como IBM MQ MQI client especificar información sobre un autenticador que se va a utilizar para la conexión de cliente. La estructura es un parámetro de entrada en la llamada MQCONNX.

Disponibilidad

La estructura MQAIR está disponible para los clientes siguientes:

- AIX
- Linux
- Solaris
- Windows

Juego de caracteres y codificación

Los datos de MQAIR deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 468. Campos en MQAIR

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQAIR_ID_ESTRUCTURA	'AIR¬'
Versión (número de versión de estructura)	MQAIR_VERSION_1	1
AuthInfoTipo (tipo de información de autenticación)	MQAIT_CRL_LDAP	1
AuthInfoConnName (nombre de conexión del servidor CRL de LDAP)	Ninguna	Serie nula o espacios en blanco
LDAPUserNamePtr (dirección del nombre de usuario LDAP)	Ninguna	Puntero nulo o bytes nulos
LDAPUserNameDesplazamiento (desplazamiento del nombre de usuario LDAP desde el inicio de MQSCO)	Ninguna	0
LDAPUserNameLongitud (longitud del nombre de usuario LDAP)	Ninguna	0
LDAPPassword (contraseña para acceder al servidor LDAP)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Versión* es menor que MQAIR_VERSION_2.		
OCSPResponderURL (URL en el que se puede contactar con el programa de respuesta OCSP)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQAIR_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */

};
```

Declaración COBOL para 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).
```

Declaración de Visual Basic para 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)

El valor debe ser:

MQAIR_ID_ESTRUCTURA
Identificador del registro de información de autenticación.

Para el lenguaje de programación C, también se define la constante MQAIR_STRUC_ID_ARRAY; tiene el mismo valor que MQAIR_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQAIR_STRUC_ID.

Versión (MQLONG)

El número de versión de la estructura MQAIR.

El valor debe ser uno de los siguientes:

MQAIR_VERSION_1
Registro de información de autenticación Version-1 .

MQAIR_VERSION_2
Registro de información de autenticación Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

VERSIÓN_ACTUAL_MQAIR_VERSIÓN
Versión actual del registro de información de autenticación.

Siempre es un campo de entrada. El valor inicial de este campo es MQAIR_VERSION_1.

Tipo AuthInfo(MQLONG)

Este es el tipo de información de autenticación contenida en el registro.

El valor puede ser uno de los dos parámetros siguientes:

MQAIT_CRL_LDAP
Comprobación de revocación de certificados utilizando el servidor LDAP.

MQAIT_OCSP
Comprobación de revocación de certificados utilizando OCSP.

Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_TYPE_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQAIT_CRL_LDAP.

AuthInfoConnName (MQCHAR264)

Es el nombre de host o la dirección de red de un host en el que se ejecuta el servidor LDAP. Esto puede ir seguido de un número de puerto opcional, entre paréntesis. El número de puerto predeterminado es 389.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_CONN_NAME_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_AUTH_INFO_CONN_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

LDAPUserNamePtr (PMQCHAR)

Es el nombre de usuario de LDAP.

Consta del nombre distinguido del usuario que está intentando acceder al servidor CRL de LDAP. Si el valor es más corto que la longitud especificada por *LDAPUserNameLength*, termine el valor con un carácter nulo, o rellena el valor con espacios en blanco hasta la longitud *LDAPUserNameLength*. El campo se ignora si *LDAPUserNameLength* es cero.

Puede proporcionar el nombre de usuario LDAP de una de estas dos maneras:

- Utilizando el campo de puntero *LDAPUserNamePtr*

  En este caso, la aplicación puede declarar una serie que esté separada de la estructura MQAIR y establecer *LDAPUserNamePtr* en la dirección de la serie.

  Considere la posibilidad de utilizar *LDAPUserNamePtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).
- Utilizando el campo de desplazamiento *LDAPUserNameOffset*

En este caso, la aplicación debe declarar una estructura compuesta que contenga la estructura MQSCO seguida de la matriz de registros MQAIR seguida de las series de nombre de usuario LDAP y establecer *LDAPUserNameOffset* en el desplazamiento de la serie de nombre adecuada desde el inicio de la estructura MQAIR. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

Considere la posibilidad de utilizar *LDAPUserNameOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que podría no ser portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Cualquiera que sea la técnica elegida, utilice sólo una de *LDAPUserNamePtr* y *LDAPUserNameOffset* ; la llamada falla con el código de razón MQRC_LDAP_USER_NAME_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

LDAPUserNameDesplazamiento (MQLONG)

Es el desplazamiento en bytes del nombre de usuario LDAP desde el inicio de la estructura MQAIR.

El desplazamiento puede ser positivo o negativo. El campo se ignora si *LDAPUserNameLength* es cero.

Puede utilizar *LDAPUserNamePtr* o *LDAPUserNameOffset* para especificar el nombre de usuario LDAP, pero no ambos; consulte la descripción del campo *LDAPUserNamePtr* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

LDAPUserNameLongitud (MQLONG)

Es la longitud, en bytes, del nombre de usuario LDAP al que se dirige el campo *LDAPUserNamePtr* o *LDAPUserNameOffset* . El valor debe estar en el rango de cero a MQ_DISTINGUISHED_NAME_LENGTH. Si el valor no es válido, la llamada falla con el código de razón MQRC_LDAP_USER_NAME_LENGTH_ERR.

Si el servidor LDAP implicado no requiere un nombre de usuario, establezca este campo en cero.

Este es un campo de entrada. El valor inicial de este campo es 0.

LDAPPassword (MQCHAR32)

Esta es la contraseña necesaria para acceder al servidor CRL de LDAP. Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo.

Si el servidor LDAP no requiere una contraseña, u omite el nombre de usuario LDAP, *LDAPPassword* debe ser nulo o estar en blanco. Si omite el nombre de usuario LDAP y *LDAPPassword* no es nulo o está en blanco, la llamada falla con el código de razón MQRC_LDAP_PASSWORD_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_LDAP_PASSWORD_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

OCSPResponderURL (MQCHAR256)

Para una estructura MQAIR que representa detalles de conexión para un programa de respuesta OCSP, este campo contiene el URL en el que se puede contactar con el programa de respuesta.

El valor de este campo es un URL HTTP. Este campo tiene prioridad sobre un URL en una extensión de certificado AIA (AuthorityInfoAccess).

El valor se ignora a menos que las dos sentencias siguientes sean verdaderas:

- La estructura MQAIR es de la versión 2 o posterior (el campo Versión se establece en MQAIR_VERSION_2 o superior).

- El campo de tipo AuthInfose establece en MQAIT_OCSP.

Si el campo no contiene un URL HTTP en el formato correcto (y no se ignora), la llamada MQCONNX falla con el código de razón MQRC_OCSP_URL_ERROR.

Este campo es sensible a las mayúsculas y minúsculas. Debe empezar con la serie http:// en minúsculas. El resto del URL puede ser sensible a las mayúsculas y minúsculas, en función de la implementación del servidor OCSP.

Este campo no está sujeto a la conversión de datos.

MQBMHO-Opciones de almacenamiento intermedio a manejador de mensajes

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se producen los manejadores de mensajes a partir de los almacenamientos intermedios. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Juego de caracteres y codificación

Los datos de MQBMHO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 469. Campos en MQBMHO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQBMHO_STRUC_ID	'BMHO'
Versión (número de versión de estructura)	MQBMHO_VERSION_1	1
Options (opciones que controlan la acción de MQBMHO)	MQBMHO_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQBMHO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQBMHO MyBMHO = {MQBMHO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración High Level Assembler para 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)

Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQBMHO_STRUC_ID
: Identificador de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

  Para el lenguaje de programación C, la constante MQBMHO_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQBMHO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_STRUC_ID.

Versión (MQLONG)

Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQBMHO_VERSION_1
: Número de versión para la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQBMHO_CURRENT_VERSION
: Versión actual de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_VERSION_1.

Opciones (MQLONG)

Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo Opciones

El valor puede ser:

MQBMHO_DELETE_PROPERTIEs
: Las propiedades que se añaden al descriptor de contexto de mensaje se suprimen del almacenamiento intermedio. Si la llamada falla, no se suprimen las propiedades.

Opciones predeterminadas: Si no necesita la opción descrita, utilice la opción siguiente:

MQBMHO_NONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_DELETE_PROPERcities.

MQBO-Opciones de inicio

La estructura MQBO permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo. La estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Disponibilidad

La estructura MQBO está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

La estructura MQBO no está disponible para IBM MQ MQI clients.

Juego de caracteres y codificación

Los datos de MQBO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 470. Campos en MQBO para MQBO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQBO_STRUC_ID	'BO¬¬'
Versión (número de versión de estructura)	MQBO_VERSION_1	1
Options (opciones que controlan la acción de MQBEGIN)	MQBO_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQBO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de Visual Basic para 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)

Este campo siempre es un campo de entrada. Su valor inicial es MQBO_STRUC_ID.

El valor debe ser:

MQBO_STRUC_ID
Identificador de la estructura de opciones de inicio.

Para el lenguaje de programación C, también se define la constante MQBO_STRUC_ID_ARRAY; tiene el mismo valor que MQBO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)

Este campo siempre es un campo de entrada. Su valor inicial es MQBO_VERSION_1.

El valor debe ser:

MQBO_VERSION_1
Número de versión para la estructura de opciones de inicio.

La constante siguiente especifica el número de versión de la versión actual:

MQBO_CURRENT_VERSION
Versión actual de la estructura de opciones de inicio.

Opciones (MQLONG)

Este campo siempre es un campo de entrada. Su valor inicial es MQBO_NONE.

El valor debe ser:

MQBO_NONE
No se ha especificado ninguna opción.

MQCBC-Contexto de devolución de llamada

La estructura MQCBC se utiliza para especificar información de contexto que se pasa a una función de devolución de llamada. La estructura es un parámetro de entrada/salida en la llamada a una rutina de consumidor de mensajes.

Disponibilidad

La estructura MQCBC está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCBC es MQCBC_VERSION_2.

Juego de caracteres y codificación

Los datos de MQCBC deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura estará en el juego de caracteres y la codificación del cliente.

Campos

No hay valores iniciales para la estructura **MQCBC** . La estructura se pasa como un parámetro a una rutina de devolución de llamada. El gestor de colas inicializa la estructura; las aplicaciones nunca la inicializan.

Notas:

- En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.
- No hay valores iniciales para la estructura MQCBC. La estructura se pasa como un parámetro a una rutina de devolución de llamada. El gestor de colas inicializa la estructura; las aplicaciones nunca la inicializan.

Tabla 471. Campos en MQCBC

Campo	Descripción
StrucID	Identificador de la estructura
Versión	Número de versión de la estructura
CallType	Por qué se ha llamado a la función

Tabla 471. Campos en MQCBC (continuación)	
Campo	**Descripción**
Hobj	Descriptor de contexto del objeto
CallbackArea	Campo para que lo utilice la función de devolución de llamada
ConnectionArea	Campo para que lo utilice la función de devolución de llamada
CompCode	Código de terminación
Razón	Código de razón
Estado	Indicación del estado del consumidor actual
DataLength	Longitud del mensaje
BufferLength	Longitud del almacenamiento intermedio de mensajes en bytes
Flags	Distintivos generales
Nota: El campo restante se ignora si la versión es menor que `MQCBC_VERSION_2`	
ReconnectDelay	Número de milisegundos antes del intento de reconexión

Declaraciones lingüísticas

Declaración C para 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 */
```

Declaración COBOL para 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 **
```

Declaración PL/I para 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 */
```

Declaración High Level Assembler para 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)

El valor de este campo es el identificador de estructura.

El valor debe ser:

MQCBC_STRUC_ID
: Identificador de la estructura de contexto de devolución de llamada.

  Para el lenguaje de programación C, también se define la constante MQCBC_STRUC_ID_ARRAY; tiene el mismo valor que MQCBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBC_STRUC_ID.

Versión (MQLONG)

El valor de este campo es el número de versión de la estructura.

El valor debe ser:

MQCBC_VERSION_1
: Estructura de contexto de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCBC_CURRENT_VERSION
Versión actual de la estructura de contexto de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBC_VERSION_1.

La función de devolución de llamada siempre se pasa a la última versión de la estructura.

CallType (MQLONG)

Campo que contiene información sobre por qué se ha llamado a esta función; se definen los valores siguientes.

Tipos de llamada de entrega de mensajes: estos tipos de llamada contienen información sobre un mensaje. Los parámetros **DataLength** y **BufferLength** son válidos para estos tipos de llamada.

MQCBCT_MSG_REMOVED
La función de consumidor de mensajes se ha invocado con un mensaje que se ha eliminado de forma destructiva del descriptor de contexto de objeto.

Si el valor de *CompCode* es MQCC_WARNING, el valor del campo *Reason* es MQRC_TRUNCATED_MSG_ACCEPTED o uno de los códigos que indican un problema de conversión de datos.

MQCBCT_MSG_NOT_REMOVED
La función de consumidor de mensajes se ha invocado con un mensaje que todavía no se ha eliminado de forma destructiva del descriptor de objeto. El mensaje se puede eliminar de forma destructiva del descriptor de objeto utilizando *MsgToken*.

Es posible que el mensaje no se haya eliminado porque:

- Las opciones MQGMO solicitaron una operación de examen, MQGMO_BROWSE_ *
- El mensaje es mayor que el almacenamiento intermedio disponible y las opciones MQGMO no especifican MQGMO_ACCEPT_TRUNCATED_MSG

Si el valor de *CompCode* es MQCC_WARNING, el valor del campo *Reason* es MQRC_TRUNCATED_MSG_FAILED o uno de los códigos que indican un problema de conversión de datos.

Tipos de llamada de control de devolución de llamada: estos tipos de llamada contienen información sobre el control de la devolución de llamada y no contienen detalles sobre un mensaje. Estos tipos de llamada se solicitan utilizando Opciones en la estructura MQCBD.

Los parámetros **DataLength** y **BufferLength** no son válidos para estos tipos de llamada.

MQCBCT_REGISTER_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración inicial.

La función de devolución de llamada se invoca inmediatamente después de que se registre la devolución de llamada, es decir, al volver de una llamada MQCB utilizando un valor para el campo *Operation* de MQOP_REGISTER.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, es la primera invocación de la función de devolución de llamada.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_START_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración cuando se inicia, por ejemplo, restableciendo los recursos que se han limpiado cuando se detuvo anteriormente.

La función de devolución de llamada se invoca cuando se inicia la conexión utilizando MQOP_START o MQOP_START_WAIT.

Si una función de devolución de llamada se registra dentro de otra función de devolución de llamada, este tipo de llamada se invoca cuando se devuelve la devolución de llamada.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_STOP_CALL

La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna limpieza cuando se detiene durante un tiempo, por ejemplo, limpiando recursos adicionales que se han adquirido durante el consumo de mensajes.

La función de devolución de llamada se invoca cuando se emite una llamada MQCTL utilizando un valor para el campo *Operation* de MQOP_STOP.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *Reason* se establece para indicar la razón de la detención.

MQCBCT_DEREGISTER_CALL

La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice la limpieza final al final del proceso de consumo. La función de devolución de llamada se invoca cuando:

- La función de devolución de llamada se anula del registro utilizando una llamada MQCB con MQOP_DEREGISTER.
- La cola está cerrada, lo que provoca un anulación de registro implícito. En esta instancia, la función de devolución de llamada se pasa a MQHO_UNUSABLE_HOBJ como descriptor de contexto de objeto.
- La llamada MQDISC se completa-causando un cierre implícito y, por lo tanto, un anulación del registro. En este caso, la conexión no se desconecta inmediatamente y cualquier transacción en curso todavía no se confirma.

Si alguna de estas acciones se realiza dentro de la propia función de devolución de llamada, la acción se invoca una vez que se devuelve la devolución de llamada.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, esta es la última invocación de la función de devolución de llamada.

El valor del campo *Reason* se establece para indicar la razón de la detención.

MQCBCT_EVENT_CALL

Función de manejador de sucesos

La función de manejador de sucesos se ha invocado sin un mensaje cuando el gestor de colas o la conexión se detiene o desactiva temporalmente.

Esta llamada se puede utilizar para realizar la acción adecuada para todas las funciones de devolución de llamada.

Función de consumidor de mensajes

La función de consumidor de mensajes se ha invocado sin un mensaje cuando se ha detectado un error (*CompCode* = MQCC_FAILED) que es específico del descriptor de contexto de objeto; por ejemplo, *Reason* code = MQRC_GET_inhibiTED.

El valor del campo *Reason* se establece para indicar la razón de la llamada.

MQCBCT_MC_EVENT_CALL

La función de manejador de sucesos se ha invocado para sucesos de multidifusión; el manejador de sucesos se envía IBM MQ Sucesos de multidifusión en lugar de sucesos 'normales' IBM MQ .

Para obtener más información sobre MQCBCT_MC_EVENT_CALL, consulte Informes de excepciones de multidifusión.

Hobj (MQHOBJ)
Es el descriptor de contexto de objeto para las llamadas al consumidor de mensajes.

Para un manejador de sucesos, este valor es MQHO_NONE

La aplicación puede utilizar este descriptor de contexto y la señal de mensaje en el bloque Obtener opciones de mensaje para obtener el mensaje si no se ha eliminado un mensaje de la cola.

Siempre es un campo de entrada. El valor inicial de este campo es MQHO_UNUSABLE_HOBJ

CallbackArea (MQPTR)
Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo CallbackArea de la estructura MQCBD, que es un parámetro de la llamada MQCB utilizada para definir la función de devolución de llamada.

Los cambios en *CallbackArea* se conservan en las invocaciones de la función de devolución de llamada para un *HObj*. Este campo no se comparte con las funciones de devolución de llamada para otros manejadores.

Es un campo de entrada/salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

ConnectionArea (MQPTR)
Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo ConnectionArea de la estructura MQCTLO, que es un parámetro de la llamada MQCTL utilizada para controlar la función de devolución de llamada.

Los cambios realizados en este campo por las funciones de devolución de llamada se conservan en las invocaciones de la función de devolución de llamada. Esta área se puede utilizar para pasar información que deben compartir todas las funciones de devolución de llamada. A diferencia de *CallbackArea*, esta área es común en todas las devoluciones de llamada para un descriptor de conexión.

Es un campo de entrada y salida. El valor inicial de este campo es un puntero nulo o bytes nulos.

CompCode (MQLONG)
Este campo es el código de terminación. Indica si se han producido problemas al consumir el mensaje.

El valor puede ser uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Este es un campo de entrada. El valor inicial de este campo es MQCC_OK.

Razón (MQLONG)
Este es el código de razón que califica el *CompCode*.

Este es un campo de entrada. El valor inicial de este campo es MQRC_NONE.

Estado (MQLONG)

Indicación del estado del consumidor actual. Este campo es de mayor valor para una aplicación cuando se pasa un código de razón distinto de cero a la función de consumidor.

Puede utilizar este campo para simplificar la programación de aplicaciones porque no es necesario codificar el comportamiento para cada código de razón.

Este es un campo de entrada. El valor inicial de este campo es MQCS_NONE

Tabla 472.

Estado	**Acción del gestor de colas**	**Valor de constante**
MQCS_NONE Este código de razón representa una llamada normal sin información de razón adicional	No; esta es la operación normal.	0
MQCS_SUSPENDED_TEMPORARY Estos códigos de razón representan condiciones temporales.	Se llama a la rutina de devolución de llamada para informar de la condición y, a continuación, se suspende. Después de un periodo de tiempo, el sistema puede volver a intentar la operación, lo que puede llevar a que se vuelva a generar la misma condición.	1
MQCS_SUSPENDED_USER_ACTION Estos códigos de razón representan condiciones en las que la devolución de llamada debe realizar una acción para resolver la condición.	El consumidor se suspende y se llama a la rutina de devolución de llamada para informar de la condición. La rutina de devolución de llamada debe resolver la condición si es posible y RESUME o cerrar la conexión.	2
MQCS_SUSPENDED Estos códigos de razón representan anomalías que impiden más devoluciones de llamada de mensaje.	El gestor de colas suspende automáticamente la función de devolución de llamada. Si se reanuda la función de devolución de llamada, es probable que vuelva a recibir el mismo código de razón.	3
MQCS_STOPPED Estos códigos de razón representan el final del consumo de mensajes.	Se entrega al manejador de excepciones y a las devoluciones de llamada que han especificado MQCBDO_STOP_CALL. No se pueden consumir más mensajes.	4

DataLength (MQLONG)

Es la longitud en bytes de los datos de aplicación del mensaje. Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

El campo DataLength contiene la longitud del mensaje, pero no necesariamente la longitud de los datos de mensaje pasados al consumidor. Puede ser que el mensaje se haya truncado. Utilice el campo ReturnedLength en el MQGMO para determinar cuántos datos se han pasado realmente al consumidor.

Si el código de razón indica que el mensaje se ha truncado, puede utilizar el campo DataLength para determinar el tamaño del mensaje real. Esto le permite determinar el tamaño del almacenamiento intermedio necesario para acomodar los datos del mensaje y, a continuación, emitir una llamada MQCB para actualizar la LongitudMaxMsg con un valor adecuado.

Si se especifica la opción MQGMO_CONVERT, el mensaje convertido podría ser mayor que el valor devuelto para DataLength. En tales casos, es probable que la aplicación tenga que emitir una llamada

MQCB para actualizar MaxMsgLength para que sea mayor que el valor devuelto por el gestor de colas para DataLength.

Para evitar problemas de truncamiento de mensajes, especifique MaxMsgde longitud como MQCBD_FULL_MSG_LENGTH. Esto hace que el gestor de colas asigne un almacenamiento intermedio para la longitud completa del mensaje después de la conversión de datos. Sin embargo, tenga en cuenta que incluso si se especifica esta opción, todavía es posible que no haya suficiente almacenamiento disponible para procesar correctamente la solicitud. Las aplicaciones siempre deben comprobar el código de razón devuelto. Por ejemplo, si no es posible asignar almacenamiento suficiente para convertir el mensaje, los mensajes se devuelven a la aplicación sin convertir.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

BufferLength (MQLONG)

Este campo es la longitud en bytes del almacenamiento intermedio de mensajes que se ha pasado a esta función.

El almacenamiento intermedio puede ser mayor que el valor de longitud MaxMsgdefinido para el consumidor y el valor de ReturnedLength en el MQGMO.

La longitud real del mensaje se proporciona en el campo DataLength .

La aplicación puede utilizar todo el almacenamiento intermedio para sus propios fines durante la duración de la función de devolución de llamada.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de excepciones.

Flags (MQLONG)

Distintivos que contienen información sobre este consumidor.

Se define la opción siguiente:

MQCBCF_READA_BUFFER_EMPTY

Este distintivo se puede devolver si una llamada MQCLOSE anterior que utilizaba la opción MQCO_QUIESCE ha fallado con un código de razón de MQRC_READ_AHEAD_MSGS.

Este código indica que se está devolviendo el último mensaje de lectura anticipada y que el almacenamiento intermedio está ahora vacío. Si la aplicación emite otra llamada MQCLOSE utilizando la opción MQCO_QUIESCE), se ejecuta correctamente.

Tenga en cuenta que no se garantiza que a una aplicación se le asigne un mensaje con este distintivo establecido, ya que es posible que todavía haya mensajes en el almacenamiento intermedio de lectura anticipada que no coincidan con los criterios de selección actuales. En este caso, la función de consumidor se invoca con el código de razón MQRC_HOBJ_QUIESCED.

Si el almacenamiento intermedio de lectura anticipada está completamente vacío, el consumidor se invoca con el distintivo MQCBCF_READA_BUFFER_EMPTY y el código de razón MQRC_HOBJ_QUIESCED_NO_MSGS.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

ReconnectDelay (MQLONG)

ReconnectDelay indica cuánto tiempo esperará el gestor de colas antes de intentar volver a conectarse. El campo puede ser modificado por un manejador de sucesos para cambiar el retardo o detener la reconexión por completo.

Utilice el campo `ReconnectDelay` sólo si el valor del campo `Razón` en el Contexto de devolución de llamada es `MQRC_RECONNECTING`.

Al entrar en el manejador de sucesos, el valor de `ReconnectDelay` es el número de milisegundos que el gestor de colas va a esperar antes de realizar un intento de reconexión. La Tabla 473 en la página 287

lista los valores que puede establecer para modificar el comportamiento del gestor de colas en el retorno del manejador de sucesos.

Tabla 473. Valores de ReconnectDelay		
Nombre	**Valor**	**Descripción**
MQRD_NO_RECONNECT	-1	No realice más intentos de reconexión. Se devuelve un error a la aplicación.
MQRD_NO_DELAY	0	Intente volver a conectarse inmediatamente.
Milliseconds	>0	Espere este número de milisegundos antes de volver a intentar la conexión.

MQCBD-Descriptor de devolución de llamada

La estructura MQCBD se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas. La estructura es un parámetro de entrada en la llamada MQCB.

Disponibilidad

La estructura MQCBD está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCBD es MQCBD_VERSION_1.

Juego de caracteres y codificación

Los datos de MQCBD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 474. Campos en MQCBD		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
StrucID (identificador de estructura)	MQCBD_ID_STRUCD	'CBD¬'
Versión (número de versión de estructura)	MQCBD_VERSION_1	1

Tabla 474. Campos en MQCBD (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
CallbackType (tipo de función de devolución de llamada)	MQCBT_MESSAGE_CONSUMER	1
Opciones (opciones que controlan el consumo de mensajes)	MQCBDO_NONE	0
CallbackArea (campo para que lo utilice la función de devolución de llamada)	Ninguna	Puntero nulo o blancos nulos
CallbackFunction (si la función se invoca como una llamada de API)	Ninguna	Puntero nulo o blancos nulos
CallbackName (si la función se invoca como un programa enlazado dinámicamente)	Ninguna	Serie nula o espacios en blanco
MaxMsgLength (longitud del mensaje más largo que se puede leer)	MQCBD_FULL_MSG_LENGTH	-1

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Cadena nula o blancos indica la cadena nula en el lenguaje de programación C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQCBD_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCBD MyCBD = {MQCBD_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
 };
```

Declaración COBOL para 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.
```

Declaración PL/I para 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)

Estructura del descriptor de devolución de llamada-Campo StrucId

Este es el identificador de estructura; el valor debe ser:

MQCBD_ID_STRUCD
Identificador de la estructura del descriptor de devolución de llamada.

Para el lenguaje de programación C, también se define la constante MQCBD_STRUC_ID_ARRAY; tiene el mismo valor que MQCBD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBD_STRUC_ID.

Versión (MQLONG)

Estructura de descriptor de devolución de llamada-Campo Versión

Este es el número de versión de la estructura; el valor debe ser:

MQCBD_VERSION_1
Estructura del descriptor de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCBD_CURRENT_VERSION
Versión actual de la estructura del descriptor de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBD_VERSION_1.

CallbackType (MQLONG)

Estructura del descriptor de devolución de llamada-Campo CallbackType

Es el tipo de la función de devolución de llamada. El valor debe ser uno de los siguientes:

MQCBT_MESSAGE_CONSUMER
Define esta devolución de llamada como una función de consumidor de mensajes.

Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto y se inicia la conexión.

MQCBT_EVENT_HANDLER
Define esta devolución de llamada como la rutina de suceso asíncrona; no se controla que consuma mensajes para un descriptor de contexto.

Hobj no es necesario en la llamada MQCB que define el manejador de sucesos y se ignora si se especifica.

Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de consumidor de mensajes. La función de consumidor se invoca sin un mensaje cuando se produce un suceso, por ejemplo, un gestor de colas o una conexión deteniéndose o desactivándose temporalmente. No se llama para condiciones que son específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INITED.

Los sucesos se entregan a la aplicación, independientemente de si la conexión se ha iniciado o detenido, excepto en los entornos siguientes:

- Entorno CICS en z/OS
- aplicaciones sin hebras

Si el llamante no pasa uno de estos valores, la llamada falla con un código *Reason* de MQRC_CALLBACK_TYPE_ERROR

Siempre es un campo de entrada. El valor inicial de este campo es MQCBT_MESSAGE_CONSUMER.

Opciones (MQLONG)

Estructura de descriptor de devolución de llamada-Campo Opciones

Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

MQCBDO_FAIL_IF_QUIESCING
La llamada MQCB falla si el gestor de colas está en estado de inmovilización.

En z/OS, esta opción también fuerza que la llamada MQCB falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

Especifique MQGMO_FAIL_IF_QUIESCING, en las opciones MQGMO pasadas en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

Opciones de control: Las opciones siguientes controlan si se llama a la función de devolución de llamada, sin un mensaje, cuando cambia el estado del consumidor:

MQCBDO_REGISTER_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_REGISTER_CALL.

MQCBDO_START_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_START_CALL.

MQCBDO_STOP_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_STOP_CALL.

MQCBDO_DEREGISTER_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_DEREGISTER_CALL.

MQCBDO_EVENT_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENT_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT__MC_EVENT_CALL.

Consulte CallType para obtener más detalles sobre estos tipos de llamada.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQCBDO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQCBDO_NONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCBDO_NONE.

CallbackArea (MQPTR)

Estructura del descriptor de devolución de llamada-Campo CallbackArea

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo CallbackArea de la estructura MQCBC, que es un parámetro de la declaración de la función de devolución de llamada.

El valor sólo se utiliza en un *Operation* que tenga un valor MQOP_REGISTER, sin ninguna devolución de llamada definida actualmente, no sustituye a una definición anterior.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CallbackFunction (MQPTR)

Estructura del descriptor de devolución de llamada-Campo CallbackFunction

La función de devolución de llamada se invoca como una llamada de función.

Utilice este campo para especificar un puntero a la función de devolución de llamada.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Si no se establece *CallbackName* ni *CallbackFunction* , la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Esta opción no está soportada en el entorno siguiente: lenguajes de programación y compiladores que no dan soporte a referencias de puntero de función. En tales situaciones, la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

z/OS En z/OS, la función debe esperar que se llame con convenios de enlace de sistema operativo. Por ejemplo, en el lenguaje de programación C, especifique:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Este es un campo de entrada. El valor inicial de este campo es un puntero nulo o bytes nulos.

Nota: Cuando se utiliza CICS con IBM WebSphere MQ 7.0.1, se da soporte al consumo asíncrono si:

- Apar PK66866 se aplica a CICS TS 3.2
- Apar PK89844 se aplica a CICS TS 4.1

CallbackName (MQCHAR128)

Estructura del descriptor de devolución de llamada-Campo CallbackName

La función de devolución de llamada se invoca como un programa enlazado dinámicamente.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Si no se establece *CallbackName* ni *CallbackFunction* , la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

El módulo se carga cuando se registra la primera rutina de devolución de llamada que se va a utilizar y se descarga cuando se anula el registro de la última rutina de devolución de llamada que se va a utilizar.

Excepto cuando se indica en el texto siguiente, el nombre se justifica por la izquierda dentro del campo, sin blancos intercalados; el propio nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

IBM i

El nombre de devolución de llamada puede tener uno de los formatos siguientes:

- Programa de biblioteca "/"

- Library "/" ServiceProgram "("FunctionName")"

Por ejemplo, `MyLibrary/MyProgram(MyFunction)`.

El nombre de biblioteca puede ser *LIBL. Los nombres de biblioteca y de programa están limitados a un máximo de 10 caracteres.

UNIX

El nombre de devolución de llamada es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[path]library(function)
```

Si no se especifica la vía de acceso, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

Windows

El nombre de devolución de llamada es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de la función debe estar entre paréntesis. El nombre de la biblioteca puede tener como prefijo opcional una ruta de directorio y una unidad:

```
[d:][path]library(function)
```

Si la unidad y la vía de acceso no se especifican, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD.

El nombre está limitado a un máximo de 8 caracteres.

z/OS CICS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro PROGRAM de la macro de mandato EXEC CICS LINK.

El nombre está limitado a un máximo de 8 caracteres.

El programa puede definirse como remoto utilizando la opción REMOTESYTEM de la definición PROGRAM instalada o mediante el programa de direccionamiento dinámico.

La región CICS remota debe estar conectada a IBM MQ si el programa va a utilizar llamadas de API de IBM MQ . Sin embargo, tenga en cuenta que el campo Hobj de la estructura MQCBC no es válido en un sistema remoto.

Si se produce una anomalía al intentar cargar *`CallbackName`*, se devuelve uno de los siguientes códigos de error a la aplicación:

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

También se graba un mensaje en el registro de errores que contiene el nombre del módulo para el que se ha intentado la carga y el código de razón de anomalía del sistema operativo.

Este es un campo de entrada. El valor inicial de este campo es una serie nula o espacios en blanco.

MaxMsgLongitud (MQLONG)

Es la longitud en bytes del mensaje más largo que se puede leer desde el descriptor de contexto y se puede asignar a la rutina de devolución de llamada. Estructura del descriptor de devolución de llamada-Campo de longitud MaxMsg

Si un mensaje tiene una longitud mayor, la rutina de devolución de llamada recibe *MaxMsgLength* bytes del mensaje y el código de razón:

- MQRC_TRUNCATED_MSG_FAILED o
- MQRC_TRUNCATED_MSG_ACCEPTED si ha especificado MQGMO_ACCEPT_TRUNCATED_MSG.

La longitud real del mensaje se proporciona en el campo DataLength de la estructura MQCBC.

Se define el siguiente valor especial:

MQCBD_FULL_MSG_LENGTH

El sistema ajusta la longitud del almacenamiento intermedio para devolver mensajes sin truncamiento.

Si no hay suficiente memoria disponible para asignar un almacenamiento intermedio para recibir el mensaje, el sistema llama a la función de devolución de llamada con un código de razón MQRC_STORAGE_NOT_AVAILABLE.

Si, por ejemplo, solicita la conversión de datos y no hay suficiente memoria disponible para convertir los datos del mensaje, el mensaje no convertido se pasa a la función de devolución de llamada.

Este es un campo de entrada. El valor inicial del campo *MaxMsgLength* es MQCBD_FULL_MSG_LENGTH.

MQCHARV-Serie de longitud variable

Utilice la estructura MQCHARV para describir una serie de longitud variable.

Disponibilidad

La estructura MQCHARV está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Juego de caracteres y codificación

Los datos de MQCHARV deben estar en la codificación del gestor de colas local que proporciona MQENC_NATIVE y el juego de caracteres del campo VSCCSID dentro de la estructura. Si la aplicación se ejecuta como un cliente MQ , la estructura debe estar en la codificación del cliente. Algunos juegos de caracteres tienen una representación que depende de la codificación. Si VSCCSID es uno de estos juegos de caracteres, la codificación utilizada es la misma que la de los otros campos de MQCHARV. El juego de caracteres identificado por VSCCSID puede ser un juego de caracteres de doble byte (DBCS).

Utilización

La estructura MQCHARV direcciona los datos que pueden no estar contiguos con la estructura que los contiene. Para abordar estos datos, se pueden utilizar los campos declarados con el tipo de datos de puntero. Tenga en cuenta que COBOL no da soporte al tipo de datos de puntero en todos los entornos.

Debido a esto, los datos también se pueden abordar utilizando campos que contienen el desplazamiento de los datos desde el inicio de la estructura que contiene MQCHARV.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 475. Campos en MQCHARV

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
VSPtr (puntero a la serie de longitud variable)	Ninguna	Puntero nulo o bytes nulos.
VSOffset (desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV)	Ninguna	0
VSBufSize (tamaño en bytes del almacenamiento intermedio direccionado por el campo VSPtr o VSOffset)	MQVS_USE_VSLENGTH	0
VSLength (longitud en bytes de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset)	Ninguna	0
VSCCSID (identificador de juego de caracteres de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset)	MQCCSI_APPL	-3

Nota: En el lenguaje de programación C, la variable de macro MQCHARV_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCHARV MyVarStr = {MQCHARV_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para MQCHARV

```
** MQCHARV structure
 10  MQCHARV.
** Address of variable length string
 15  MQCHARV-VSPTR        POINTER.
** Offset of variable length string
 15  MQCHARV-VSOFFSET     PIC S9(9) BINARY.
** Size of buffer
 15  MQCHARV-VSBUFSIZE    PIC S9(9) BINARY.
** Length of variable length string
 15  MQCHARV-VSLENGTH     PIC S9(9) BINARY.
```

```
** CCSID of variable length string
 15  MQCHARV-VSCCSID      PIC S9(9) BINARY.
```

Nota: Si desea portar una aplicación COBOL entre entornos, debe averiguar si el tipo de datos de puntero está disponible en todos los entornos previstos. Si no es así, la aplicación debe direccionar los datos utilizando los campos de desplazamiento en lugar de los campos de puntero. En entornos en los que los punteros no están soportados, puede declarar los campos de puntero como series de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes all-null. No modifique este valor inicial si está utilizando los campos de desplazamiento. Una forma de hacerlo sin cambiar los libros de copia suministrados es utilizar lo siguiente:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

donde CMQCHRVV se puede intercambiar para que se utilice el libro de copias.

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Es un puntero a la serie de longitud variable.

Puede utilizar el campo VSPtr o VSOffset para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es un puntero nulo o bytes nulos.

VSOffset (MQLONG)

El desplazamiento puede ser positivo o negativo. Puede utilizar el campo VSPtr o VSOffset para especificar la serie de longitud variable, pero no ambos. Desplazamiento en bytes de la serie de longitud variable desde el inicio de MQCHARV, o la estructura que la contiene.

Cuando la estructura MQCHARV está incorporada en otra estructura, este valor es el desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV. Cuando la estructura MQCHARV no está incorporada dentro de otra estructura, por ejemplo, si se especifica como parámetro en una llamada de función, el desplazamiento es relativo al inicio de la estructura MQCHARV.

El valor inicial de este campo es 0.

VSBufSize (MQLONG)

Es el tamaño en bytes del almacenamiento intermedio direccionado por el campo VSPtr o VSOffset.

Cuando se utiliza la estructura MQCHARV como campo de salida en una llamada a función, este campo debe inicializarse con la longitud del almacenamiento intermedio proporcionado. Si el valor de

VSLength es mayor que VSBufSize , sólo se devuelven VSBufSize bytes de datos al interlocutor en el almacenamiento intermedio.

Este valor debe ser un valor mayor o igual que cero, o el siguiente valor especial que se reconoce:

MQVS_USE_VSLENGTH
Cuando se especifica, la longitud del almacenamiento intermedio se toma del campo VSLength en la estructura MQCHARV. No utilice este valor cuando utilice la estructura como campo de salida y se proporcione un almacenamiento intermedio.

Este es el valor inicial de este campo.

VSLength (MQLONG)

Longitud en bytes de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset.

El valor inicial de este campo es 0. El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

MQVS_NULL_TERMINATED

Si no se especifica MQVS_NULL_TERMINATED, los bytes VSLength se incluyen como parte de la serie. Si hay caracteres nulos, no delimitan la serie.

Si se especifica MQVS_NULL_TERMINATED, la serie está delimitada por el primer nulo encontrado en la serie. El propio nulo no se incluye como parte de esa serie.

Nota: El carácter nulo utilizado para terminar una serie si se especifica MQVS_NULL_TERMINATED es un valor nulo del conjunto de códigos especificado por VSCCSID.

Por ejemplo, en UTF-16 (CCSID 1200, 13488 y 17584), es la codificación Unicode de dos bytes donde un valor nulo se representa mediante un número de 16 bits de todos los ceros. En UTF-16 es común encontrar bytes únicos establecidos en cero que forman parte de caracteres (por ejemplo, caracteres ASCII de 7 bits), pero las series sólo terminarán en nulo cuando se encuentren dos bytes 'cero' en un límite de bytes par. Es posible obtener dos 'cero' bytes en un límite impar cuando cada uno forma parte de caracteres válidos. Por ejemplo, x '01' x '00 x' 00 'x' 30 ' representa dos caracteres Unicode válidos y no termina de forma nula la serie.

VSCCSID (MQLONG)

Es el identificador de juego de caracteres de la serie de longitud variable a la que se dirige el campo **VSPtr** o **VSOffset** .

El valor inicial de este campo es *MQCCSI_APPL*, definido por MQ para indicar que debe cambiarse al identificador de juego de caracteres verdadero del proceso actual. Como resultado, el valor de la constante *MQCCSI_APPL* nunca se asocia con una serie de longitud variable.

El valor inicial de este campo se puede cambiar definiendo un valor diferente para la constante *MQCCSI_APPL* para la unidad de compilación. La forma de hacerlo depende del lenguaje de programación de la aplicación.

z/OS En sistemas z/OS , la aplicación predeterminada `CCSID` utilizada por *MQCCSI_APPL* se define de la forma siguiente:

- Para las aplicaciones LE por lotes que utilizan la interfaz DLL, el valor predeterminado es `CODESET` asociado con el entorno local actual en el momento en que se emite **MQCONN** (el valor predeterminado es 1047).
- Para las aplicaciones LE por lotes enlazadas con uno de los apéndices de MQ por lotes, el valor predeterminado es `CODESET` asociado con el entorno local actual en el momento de la primera llamada MQI emitida después de **MQCONN** (el valor predeterminado es 1047).
- Para las aplicaciones no LE por lotes que se ejecutan en una hebra USS, el valor predeterminado es el valor de `THLICCSID` en el momento de la primera llamada MQI emitida después de **MQCONN** (el valor predeterminado es 1047).
- Para otras aplicaciones por lotes, el valor predeterminado es el `CCSID` del gestor de colas.

Redefinición de MQCCSI_APPL

Los ejemplos siguientes muestran cómo puede alterar temporalmente el valor de MQCCSI_APPL en varios lenguajes de programación. Puede cambiar el valor de MQCCSI_APPL, eliminando la necesidad de establecer el VSCCSID para cada serie de longitud variable por separado. En estos ejemplos, el CCSID se establece en 1208; cámbielo por el valor que necesite. Esto se convierte en el valor predeterminado, que puede alterar temporalmente estableciendo el VSCCSID en cualquier instancia específica de MQCHARV.

Uso de C

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

Utilización de COBOL

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

Uso de PL/I

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

Uso de High Level Assembler

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

MQCIH-Cabecera CICS bridge

La estructura MQCIH describe la información de cabecera para un mensaje enviado a CICS a través de CICS bridge.

Para cualquier plataforma soportada por IBM MQ , puede crear y transmitir un mensaje que incluya la estructura MQCIH, pero sólo un gestor de colas IBM MQ for z/OS puede utilizar CICS bridge. Por lo tanto, para que el mensaje llegue a CICS desde un gestor de colas noz/OS , la red de gestores de colas debe incluir al menos un gestor de colas z/OS a través del cual se pueda direccionar el mensaje.

Todas las versiones de CICS soportadas por IBM MQ 9.0.0y posteriores utilizan la versión proporcionada por CICS del puente. Para obtener más información sobre cómo configurar el adaptador de IBM MQ CICS y los componentes de IBM MQ CICS bridge, consulte la sección Configuración de conexiones con MQ de la documentación de CICS.

Disponibilidad

La estructura MQCIH está disponible en las plataformas siguientes:

- AIX
- Linux
- Solaris
- Windows
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

MQFMT_CICS

Versión

La versión actual de MQCIH es MQCIH_VERSION_2. Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQCIH, con el valor inicial del campo *Version* establecido en MQCIH_VERSION_2.

Juego de caracteres y codificación

Se aplican condiciones especiales al juego de caracteres y la codificación utilizados para la estructura MQCIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola CICS bridge deben proporcionar una estructura MQCIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQCIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQCIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola CICS bridge convierte la estructura MQCIH.
- Los datos del mensaje de aplicación que siguen a la estructura MQCIH deben estar en el mismo juego de caracteres y codificación que la estructura MQCIH. No puede utilizar los campos *CodedCharSetId* y *Encoding* en la estructura MQCIH para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

  Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 476. Campos en MQCIH para MQCIH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQCIH_STRUC_ID	'CIH¬'
Versión (número de versión de estructura)	MQCIH_VERSION_2	2
StrucLength (longitud de la estructura MQCIH)	MQCIH_LENGTH_2	180
Codificación (reservado)	Ninguna	0
CodedCharSetId (reservado)	Ninguna	0
Format (nombre de formatoMQ de los datos que siguen a MQCIH)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQCIH_NONE	0
ReturnCode (código de retorno del puente)	MQCRC_OK	0
CompCode (código de terminaciónMQ o CICS EIBRESP)	MQCC_OK	0

Tabla 476. Campos en MQCIH para MQCIH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
Razón (MQ código de razón o de comentarios, o CICS EIBRESP2)	MQRC_NONE	0
UOWControl (control de unidad de trabajo)	MQCUOWC_ONLY	273
GetWaitInterval (intervalo de espera para la llamada MQGET emitida por la tarea de puente)	MQCGWI_DEFAULT	-2
LinkType (tipo de enlace)	MQCLT_PROGRAM	1
OutputDataLength (longitud de datos COMMAREA de salida)	MQCODL_AS_INPUT	-1
FacilityKeepTime (tiempo de liberación del recurso de puente)	Ninguna	0
ADSDescriptor (descriptor ADS de envío/recepción)	MQCADSD_NONE	0
ConversationalTask (si la tarea puede ser conversacional)	MQCCT_NO	0
Estado deTaskEnd (estado al final de la tarea)	MQCTES_NOSYNC	0
Recurso (señal de recurso de puente)	MQCFAC_NONE	Nulos
Función (nombre de llamada deMQ o función EIBFN de CICS)	MQCFUNC_NONE	Espacios en blanco
AbendCode (código de terminación anómala)	Ninguna	Espacios en blanco
Autenticador (contraseña o passticket)	Ninguna	Espacios en blanco
Reserved1 (reservado)	Ninguna	Espacios en blanco
FormatoReplyTo (nombre de formatoMQ del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
RemoteSysId (ID de sistema CICS remoto a utilizar)	Ninguna	Espacios en blanco
RemoteTransId (CICS RTRANSID a utilizar)	Ninguna	Espacios en blanco
TransactionId (transacción a adjuntar)	Ninguna	Espacios en blanco
FacilityLike (atributos emulados de terminal)	Ninguna	Espacios en blanco
AttentionId (tecla AID)	Ninguna	Espacios en blanco
StartCode (código de inicio de transacción)	MQCSC_NONE	Espacios en blanco
CancelCode (código de transacción de terminación anómala)	Ninguna	Espacios en blanco
NextTransactionId (siguiente transacción a adjuntar)	Ninguna	Espacios en blanco
Reserved2 (reservado)	Ninguna	Espacios en blanco
Reserved3 (reservado)	Ninguna	Espacios en blanco
Nota: Los campos restantes no están presentes si *Version* es menor que MQCIH_VERSION_2.		
CursorPosition (posición del cursor)	Ninguna	0

Tabla 476. Campos en MQCIH para MQCIH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
ErrorOffset (desplazamiento del error en el mensaje)	Ninguna	0
InputItem (elemento de entrada)	Ninguna	0
Reserved4 (reservado)	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCIH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración High Level Assembler para 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)
```

Declaración de Visual Basic para 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
```

Utilización

Si la aplicación requiere valores que son los mismos que los valores iniciales que se muestran en Tabla 476 en la página 298, y el puente se ejecuta con AUTH=LOCAL o AUTH=IDENTIFY, puede omitir la estructura MQCIH del mensaje. En todos los demás casos, la estructura debe estar presente.

El puente acepta una estructura MQCIH version-1 o version-2 , pero para las transacciones 3270, debe utilizar una estructura version-2 .

La aplicación debe asegurarse de que los campos documentados como campos de solicitud tengan los valores adecuados en el mensaje enviado al puente; estos campos se introducen en el puente.

Los campos documentados como campos de respuesta los establece CICS bridge en el mensaje de respuesta que el puente envía a la aplicación. La información de error se devuelve en los campos *ReturnCode*, *Function*, *CompCode*, *Reason*y *AbendCode* , pero no todos se establecen en todos los casos. La tabla siguiente muestra qué campos se establecen para distintos valores de *ReturnCode*.

Tabla 477. Contenido de los campos de información de error en la estructura MQCIH para MQCIH

ReturnCode	***Function***	***CompCode***	***Reason***	***AbendCode***
MQCRC_OK	-	-	-	-
ERROR DE MQCRC_BRIDGE_	-	-	MQFB_CICS_ *	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	Nombre de llamada de MQ	MQ *CompCode*	MQ *Reason*	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	ABCODE de CICS

StrucId (MQCHAR4)

Este campo es un campo de solicitud, con un valor inicial de MQCIH_STRUC_ID.

El valor debe ser:

MQCIH_STRUC_ID
Identificador de la estructura de cabecera de información de CICS .

Para el lenguaje de programación C, también se define la constante MQCIH_STRUC_ID_ARRAY; tiene el mismo valor que MQCIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCIH_VERSION_2.

El valor debe ser uno de los siguientes:

MQCIH_VERSION_1
Version-1 CICS estructura de cabecera de información.

MQCIH_VERSION_2
Version-2 CICS estructura de cabecera de información.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de CICS .

StrucLength (MQLONG)

Este campo es un campo de solicitud, con un valor inicial de MQCIH_LENGTH_2.

El valor debe ser uno de los siguientes:

MQCIH_LENGTH_1
Longitud de la estructura de cabecera de información version-1 CICS .

MQCIH_LENGTH_2
Longitud de la estructura de cabecera de información de version-2 CICS .

La constante siguiente especifica la longitud de la versión actual:

MQCIH_CURRENT_LENGTH
Longitud de la versión actual de la estructura de cabecera de información de CICS .

Encoding (MQLONG)

Este campo es un campo reservado; su valor no es significativo. Su valor inicial es 0.

La codificación para las estructuras soportadas que siguen a una estructura MQCIH es la misma que la codificación de la propia estructura MQCIH y se toma de cualquier cabecera IBM MQ anterior.

CodedCharSetId (MQLONG)

CodedCharSetId es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

El ID de juego de caracteres para las estructuras soportadas que siguen a una estructura MQCIH es el mismo que el ID de juego de caracteres de la propia estructura MQCIH y se toma de cualquier cabecera IBM MQ anterior.

Format (MQCHAR8)

Este campo muestra el nombre de formato IBM MQ de los datos que siguen a la estructura MQCIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

Este nombre de formato también se utiliza para el mensaje de respuesta, si el campo *ReplyToFormat* tiene el valor MQFMT_NONE.

- Para las solicitudes DPL, *Format* debe ser el nombre de formato de COMMAREA.
- Para las solicitudes 3270, *Format* debe ser CSQCBDCIy el puente establece el formato en CSQCBDCO para los mensajes de respuesta.

Las salidas de conversión de datos para estos formatos deben estar instaladas en el gestor de colas donde se van a ejecutar.

Si el mensaje de solicitud genera un mensaje de respuesta de error, el mensaje de respuesta de error tiene un nombre de formato de MQFMT_STRING.

Este campo es un campo de solicitud. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Este campo es un campo de solicitud. El valor inicial de este campo es MQCIH_NONE.

El valor debe ser:

MQCIH_NONE
Sin distintivos.

MQCIH_PASS_EXPIRATION
El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud.
- El tiempo de caducidad restante del mensaje de solicitud sin realizar ningún ajuste para el tiempo de proceso del puente.

Si omite este valor, la hora de caducidad se establece en *unlimited*.

MQCIH_REPLY_WITHOUT_NULLS
La longitud del mensaje de respuesta de una solicitud de programa DPL de CICS se ajusta para excluir los nulos finales (X'00 ') al final de la COMMAREA devuelta por el programa DPL. Si no se establece este valor, es posible que los valores nulos sean significativos y se devuelva el COMMAREA completo.

MQCIH_SYNC_ON_RETURN
El enlace CICS para solicitudes DPL utiliza la opción SYNCONRETURN, lo que hace que CICS tome un punto de sincronización cuando el programa se completa si se envía a otra región CICS . El puente no especifica a qué región CICS enviar la solicitud; esto lo controla la definición de programa CICS o los recursos de equilibrio de carga de trabajo.

ReturnCode (MQLONG)

El valor de este campo es el código de retorno del CICS bridge que describe el resultado del proceso realizado por el puente. Este campo es un campo de respuesta, con un valor inicial de MQCRC_OK.

Los campos *Function*, *CompCode*, *Reason*y *AbendCode* pueden contener información adicional (consulte Tabla 477 en la página 304). El valor puede ser uno de los siguientes:

MQCRC_APPLICATION_ABEND
(5, X'005 ') La aplicación ha finalizado de forma anómala.

MQCRC_BRIDGE_ABEND
(4, X'004 ') CICS bridge ha finalizado de forma anómala.

ERROR DE MQCRC_BRIDGE_
(3, X'003 ') CICS bridge ha detectado un error.

MQCRC_BRIDGE_TIMEOUT
(8, X'008 ') Segundo o posterior mensaje dentro de la unidad de trabajo actual no recibido dentro del tiempo especificado.

MQCRC_CICS_ERROR _EXEC_ERROR
(1, X'001 ') La sentencia EXEC CICS ha detectado un error.

MQCRC_MQ_API_ERROR
(2, X'002 ') La llamada MQ ha detectado un error.

MQCRC_OK
(0, X'000 ') Sin error.

MQCRC_PROGRAM_NOT_AVAILABLE
(7, X'007 ') Programa no disponible.

MQCRC_SECURITY_ERROR
(6, X'006 ') Se ha producido un error de seguridad.

MQCRC_TRANSID_NOT_AVAILABLE
(9, X'009 ') Transacción no disponible.

CompCode (MQLONG)

Este campo es un campo de respuesta. Su valor inicial es MQCC_OK

El valor devuelto en este campo depende de *ReturnCode* ; consulte Tabla 477 en la página 304.

Razón (MQLONG)

Este campo es un campo de respuesta. Su valor inicial es MQRC_NONE.

El valor devuelto en este campo depende de *ReturnCode* ; consulte Tabla 477 en la página 304.

UOWControl (MQLONG)

Este campo es un campo de solicitud que controla el proceso de unidad de trabajo realizado por CICS bridge. El valor inicial de este campo es MQCUOWC_ONLY.

Puede solicitar al puente que ejecute una sola transacción o uno o varios programas dentro de una unidad de trabajo. El campo indica si el CICS bridge inicia una unidad de trabajo, realiza la función solicitada dentro de la unidad de trabajo actual o finaliza la unidad de trabajo comprometiéndola o restituyéndola. Se soportan varias combinaciones, para optimizar los flujos de transmisión de datos.

El valor debe ser uno de los siguientes:

MQCUOWC_ONLY
Inicie la unidad de trabajo, realice la función y, a continuación, confirme la unidad de trabajo.

MQCUOWC_CONTINUE
Datos adicionales para la unidad de trabajo actual (sólo 3270).

MQCUOWC_FIRST
Inicie la unidad de trabajo y realice la función.

MQCUOWC_MIDDLE
Realizar función dentro de la unidad de trabajo actual

MQCUOWC_LAST
Realice la función y, a continuación, confirme la unidad de trabajo.

MQCUOWC_COMMIT
Confirme la unidad de trabajo (sólo DPL).

MQCUOWC_BACKOUT
Restituir la unidad de trabajo (sólo DPL).

Intervalo GetWait(MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCGWI_DEFAULT.

Este campo sólo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST. Permite a la aplicación emisora especificar el tiempo aproximado en milisegundos que las llamadas MQGET emitidas por el puente esperarán el segundo y los mensajes de solicitud subsiguientes para la unidad de trabajo iniciada por este mensaje. Este recurso altera temporalmente el intervalo de espera predeterminado utilizado por el puente. Puede utilizar los siguientes valores especiales:

MQCGWI_DEFAULT
Intervalo de espera predeterminado.

Este valor hace que el CICS bridge espere el tiempo especificado cuando se inició el puente.

MQWI_UNLIMITED
Intervalo de espera ilimitado.

LinkType (MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCLT_PROGRAM.

Este valor indica el tipo de objeto que el puente intenta enlazar. Tiene que ser uno de los valores siguientes:

MQCLT_PROGRAM
Programa DPL.

TRANSACCIÓN_MQC
Transacción 3270.

Longitud de OutputData(MQLONG)

Este campo es un campo de petición utilizado sólo para programas DPL. Su valor inicial es MQCODL_AS_INPUT.

Este valor es la longitud de los datos de usuario que deben devolverse al cliente en un mensaje de respuesta. Esta longitud incluye el nombre de programa de 8 bytes. La longitud de COMMAREA pasada al programa enlazado es el máximo de este campo y la longitud de los datos de usuario en el mensaje de solicitud, menos 8.

Nota: La longitud de los datos de usuario en un mensaje es la longitud del mensaje excluyendo la estructura MQCIH.

Si la longitud de los datos de usuario en el mensaje de solicitud es menor que *OutputDataLength*, se utiliza la opción DATALENGTH del mandato LINK , lo que permite que LINK se envíe de forma eficaz a otra región CICS .

Puede utilizar el siguiente valor especial:

MQCODL_AS_INPUT
La longitud de salida es la misma que la longitud de entrada.

Este valor puede ser necesario incluso si no se solicita ninguna respuesta, para asegurarse de que el COMMAREA pasado al programa enlazado tiene un tamaño suficiente.

Tiempo de FacilityKeep(MQLONG)

FacilityKeepTiempo es el tiempo en segundos que se mantiene el recurso de puente después de que finalice la transacción de usuario.

Para transacciones pseudoconversacionales, especifique un valor que corresponda a la duración esperada de una pseudoconversación; especifique cero para la última transacción de una pseudoconversación, y para otros tipos de transacción especifique cero.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0.

ADSDescriptor (MQLONG)

Este campo es un indicador que especifica si se deben enviar descriptores ADS en solicitudes SEND y RECEIVE BMS.

Los valores siguientes están definidos:

MQCADSD_NONE
No envíe ni reciba descriptores ADS.

MQCADSD_SEND
Enviar descriptores ADS.

MQCADSD_RECV
Recibir descriptores ADS.

MQCADSD_MSGFORMAT
Utilice el formato de mensaje para los descriptores ADS.

Esto envía o recibe los descriptores ADS utilizando la forma larga del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

Establezca el campo *ADSDescriptor* como se indica a continuación:

- Si no utiliza descriptores ADS, establezca el campo en MQCADSD_NONE.
- Si utiliza descriptores ADS con el *mismo* CCSID en cada entorno, establezca el campo en la suma de MQCADSD_SEND y MQCADSD_RECV.
- Si utiliza descriptores ADS con CCSID *diferentes* en cada entorno, establezca el campo en la suma de MQCADSD_SEND, MQCADSD_RECV y MQCADSD_MSGFORMAT.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es MQCADSD_NONE.

ConversationalTask (MQLONG)

Este campo es un indicador que especifica si se debe permitir que la tarea emita solicitudes para obtener más información, o si se debe detener la tarea y emitir un mensaje de terminación anómala.

El valor debe ser una de las opciones siguientes:

MQCCT_YES
La tarea es conversacional.

MQCCT_NO
La tarea no es conversacional.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es MQCCT_NO.

Estado de TaskEnd(MQLONG)

Este campo es un campo de respuesta, que muestra el estado de la transacción de usuario al final de la tarea. El campo sólo se utiliza para transacciones 3270 y su valor inicial es MQCTES_NOSYNC.

Se devuelve uno de los siguientes valores:

MQCTES_NOSYNC
No sincronizado.

La transacción de usuario todavía no se ha completado y no se ha sincronizado. El campo *MsgType* en MQMD es MQMT_REQUEST en este caso.

MQCTES_COMMIT
Confirmar unidad de trabajo.

La transacción de usuario todavía no se ha completado, pero ha sincronizado la primera unidad de trabajo. El campo *MsgType* en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_BACKOUT
Restituir unidad de trabajo.

La transacción de usuario todavía no se ha completado. La unidad de trabajo actual se restituye. El campo *MsgType* en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_ENDTASK
Finalizar tarea.

La transacción de usuario ha finalizado (o ha terminado de forma anómala). El campo *MsgType* en MQMD es MQMT_REPLY en este caso.

Recurso (MQBYTE8)

Este campo muestra la señal de recurso de puente de 8 bytes.

Una señal de recurso de puente permite que varias transacciones en una pseudoconversación utilicen el mismo recurso de puente (terminal 3270 virtual). En el primer mensaje, o sólo en una pseudoconversación, establezca un valor de MQCFAC_NONE. Este valor indica a CICS que asigne un nuevo recurso de puente para este mensaje. Se devuelve una señal de recurso de puente en los mensajes de respuesta cuando se especifica un *FacilityKeepTime* distinto de cero en el mensaje de entrada. Los mensajes de entrada posteriores dentro de una pseudoconversación deben utilizar la misma señal de recurso de puente.

Se define el siguiente valor especial:

MQCFAC_NONE
No se ha especificado ninguna señal de recurso.

Para el lenguaje de programación C, la constante MQCFAC_NONE_ARRAY también está definida y tiene el mismo valor que MQCFAC_NONE, pero es una matriz de caracteres en lugar de una serie.

Este campo es a la vez un campo de solicitud y un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_FACILITY_LENGTH. El valor inicial de este campo es MQCFAC_NONE.

Función (MQCHAR4)

Este campo es un campo de respuesta. La longitud de este campo la proporciona MQ_FUNCTION_LENGTH. El valor inicial de este campo es MQCFUNC_NONE.

El valor devuelto en este campo depende de *ReturnCode* ; consulte Tabla 477 en la página 304. Los valores siguientes son posibles cuando *Function* contiene un nombre de llamada IBM MQ :

MQCFUNC_MQCONN
Llamada MQCONN.

MQCFUNC_MQGET
Llamada MQGET.

MQCFUNC_MQINQ
Llamada MQINQ.

MQCFUNC_MQOPEN
Llamada MQOPEN.

MQCFUNC_MQPUT
Llamada MQPUT.

MQCFUNC_MQPUT1
Llamada MQPUT1 .

MQCFUNC_NONE
Sin llamada.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCFUNC_ * _ARRAY; estas constantes tienen los mismos valores que las constantes MQCFUNC_ * correspondientes, pero son matrices de caracteres en lugar de series.

AbendCode (MQCHAR4)

AbendCode es un campo de respuesta. La longitud de este campo la proporciona MQ_ABEND_CODE_LENGTH. El valor inicial de este campo es de 4 caracteres en blanco.

El valor devuelto en este campo sólo es significativo si el campo *ReturnCode* tiene el valor MQCRC_APPLICATION_ABEND o MQCRC_BRIDGE_ABEND. Si lo hace, *AbendCode* contiene el valor ABCODE de CICS .

Autenticador (MQCHAR8)

El valor de este campo es la contraseña o el pase.

Si la autenticación de identificador de usuario está activa para CICS bridge, se utiliza *Authenticator* con el identificador de usuario en el contexto de identidad MQMD para autenticar el remitente del mensaje.

Este es un campo de solicitud. La longitud de este campo la proporciona MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es de 8 blancos.

Reserved1 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Formato ReplyTo(MQCHAR8)

El valor de este campo es el nombre de formato IBM MQ del mensaje de respuesta que se envía en respuesta al mensaje actual.

Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

Este campo es un campo de petición utilizado sólo para programas DPL. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

RemoteSysId (MQCHAR4)

Este campo muestra el identificador del sistema CICS del sistema CICS que procesa la solicitud.

Si este campo está en blanco, la solicitud del sistema CICS se procesa en el mismo sistema CICS que el supervisor de puente. El SYSID utilizado se devuelve en el mensaje de respuesta.

Para una pseudoconversación 3270, todos los mensajes posteriores de la conversación deben especificar el SYSID remoto devuelto en la respuesta inicial. Si se especifica, el SYSID debe:

- Estar activo.
- Tenga acceso a la cola de solicitudes de IBM MQ .
- Los enlaces ISC de CICS pueden acceder a ellos desde el sistema CICS del supervisor de puente.

RemoteTransId (MQCHAR4)

Este campo es un campo de solicitud opcional. La longitud de este campo la proporciona MQ_TRANSACTION_ID_LENGTH.

Si se especifica, el campo se utiliza como el valor RTRANSID de CICS START.

TransactionId (MQCHAR4)

Este campo es un campo de solicitud. Su longitud la proporciona MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

Si *LinkType* tiene el valor MQCLT_TRANSACTION, *TransactionId* es el identificador de transacción de la transacción de usuario que se va a ejecutar; especifique un valor no en blanco en este caso.

Si *LinkType* tiene el valor MQCLT_PROGRAM, *TransactionId* es el código de transacción bajo el que se van a ejecutar todos los programas de la unidad de trabajo. Si especifica un valor en blanco, se utiliza el código de transacción predeterminado del puente CICS DPL (CKBP). Si el valor no está en blanco, debe haberlo definido en CICS como una transacción local con un programa inicial que sea CSQCBP00. Este campo sólo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

FacilityLike (MQCHAR4)

FacilityLike es el nombre de un terminal instalado que se va a utilizar como modelo para el recurso de puente.

Un valor de blancos significa que *FacilityLike* se toma de la definición de perfil de transacción de puente o que se utiliza un valor por omisión.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_FACILITY_LIKE_LENGTH. El valor inicial de este campo es de cuatro blancos.

AttentionId (MQCHAR4)

El valor de este campo determina el valor inicial de la tecla AID cuando se inicia la transacción. Es un valor de 1 byte, alineado a la izquierda.

AttentionId es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_ATTENTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

StartCode (MQCHAR4)

El valor de este campo es un indicador que especifica si el puente emula una transacción de terminal o una transacción iniciada con START.

El valor debe ser uno de los siguientes:

MQCSC_START
: Inicio.

MQCSC_STARTDATA
: Datos de inicio.

MQCSC_TERMINPUT
: Entrada de terminal.

MQCSC_NONE
: Ninguno.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCSC_ * _ARRAY; estas constantes tienen los mismos valores que las constantes MQCSC_ * correspondientes, pero son matrices de caracteres en lugar de series.

En la respuesta del puente, este campo se establece en el código de inicio adecuado para el siguiente ID de transacción contenido en el campo *NextTransactionId* . Los siguientes códigos de inicio son posibles en la respuesta:

- MQCSC_START
- MQCSC_STARTDATA

- MQCSC_TERMINPUT

Para CICS Transaction Server 1.2, este campo es sólo un campo de solicitud; su valor en la respuesta no está definido.

Para CICS Transaction Server 1.3 y releases posteriores, este campo es tanto una solicitud como un campo de respuesta.

Este campo sólo se utiliza para transacciones 3270. La longitud de este campo la proporciona MQ_START_CODE_LENGTH. El valor inicial de este campo es MQCSC_NONE.

CancelCode (MQCHAR4)

El valor de este campo es el código de terminación anómala que se utilizará para terminar la transacción (normalmente una transacción conversacional que solicita más datos). De lo contrario, este campo se establece en blancos.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_CANCEL_CODE_LENGTH. El valor inicial de este campo es de cuatro blancos.

NextTransactionId (MQCHAR4)

Este valor es el nombre de la siguiente transacción devuelta por la transacción de usuario (normalmente por EXEC CICS RETURN TRANSID). Si no hay ninguna transacción siguiente, este campo se establece en blancos.

Este campo es un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

Reserved2 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Reserved3 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

CursorPosition (MQLONG)

El valor de este campo muestra la posición inicial del cursor cuando se inicia la transacción. Para transacciones conversacionales, la posición del cursor está en el vector RECEIVE.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

ErrorOffset (MQLONG)

El campo ErrorOffset muestra la posición de los datos no válidos detectados por la salida del puente. Este campo proporciona el desplazamiento desde el inicio del mensaje hasta la ubicación de los datos no válidos.

ErrorOffset es un campo de respuesta utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

InputItem (MQLONG)

Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

Reserved4 (MQLONG)

Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

MQCMHO-Crear opciones de manejador de mensajes

La estructura **MQCMHO** permite a las aplicaciones especificar opciones que controlan cómo se crean los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada **MQCRTMH** .

Disponibilidad

La estructura **MQCMHO** está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y con clientes IBM MQ .

Juego de caracteres y codificación

Los datos de **MQCMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 478. Campos en MQCMHO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQCMHO_STRUC_ID	'CMHO'
Versión (número de versión de estructura)	MQCMHO_VERSION_1	1
Options (opciones)	MQCMHO_DEFAULT_VALIDATION	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQCMHO_DEFAULT contiene los valores listados en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQCMHO

```
struct tagMQCMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
```

```
  MQLONG    Options;         /* Options that control the action of MQCRTMH */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_STRUC_ID.

Este es el identificador de estructura; el valor debe ser:

MQCMHO_STRUC_ID
Identificador para la estructura de opciones de creación de manejadores de mensajes.

Para el lenguaje de programación C, la constante **MQCMHO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQCMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)

Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_VERSION_1.

Este es el número de versión de la estructura; el valor debe ser:

MQCMHO_VERSION_1
Version-1 crea la estructura de opciones de manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

MQCMHO_CURRENT_VERSION
Versión actual de la estructura de opciones de creación de manejadores de mensajes.

Opciones (MQLONG)

Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_DEFAULT_VALIDATION.

Se puede especificar una de las opciones siguientes:

MQCMHO_VALIDAR

Cuando se llama a **MQSETMP** para establecer una propiedad en este descriptor de mensaje, el nombre de propiedad se valida para asegurarse de que:

- no contiene caracteres no válidos.
- no empieza JMS ni usr.JMS excepto para lo siguiente:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Estos nombres están reservados para las propiedades JMS .
- no es una de las siguientes palabras clave, en cualquier mezcla de mayúsculas o minúsculas:
  - Y
  - BETWEEN
  - ESCAPE
  - FALSE
  - IN
  - IS
  - LIKE
  - NO
  - NULL
  - OR
  - TRUE
- no empieza el cuerpo. o raíz. (excepto para Root.MQMD.).

Si la propiedad es MQ-defined (mq. *) y el nombre se reconoce, los campos de descriptor de propiedad se establecen en los valores correctos para la propiedad. Si la propiedad no se reconoce, el campo *Support* del descriptor de propiedad se establece en **MQPD_OPTIONAL**.

MQCMHO_DEFAULT_VALIDATION

Este valor especifica que se produce el nivel predeterminado de validación de los nombres de propiedad.

El nivel predeterminado de validación es equivalente al nivel especificado por **MQCMHO_VALIDATE**.

Este es el valor predeterminado.

MQCMHO_NO_VALIDATION

No se produce ninguna validación en el nombre de propiedad. Consulte la descripción de **MQCMHO_VALIDATE**.

Opción predeterminada: Si ninguna de las opciones anteriores descritas es necesaria, se puede utilizar la opción siguiente:

MQCMHO_NONE

Todas las opciones asumen sus valores predeterminados. Utilice este valor para indicar que no se ha especificado ninguna otra opción. **MQCMHO_NONE** ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

MQCNO - Opciones de conexión

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión con el gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONNX.

Para obtener más información sobre cómo utilizar manejadores compartidos y la llamada MQCONNX, consulte Conexiones compartidas (independientes de hebra) con MQCONNX.

Disponibilidad

Todas las versiones de la estructura MQCNO, excepto MQCNO_VERSION_4, están disponibles en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQCNO, pero con el valor inicial del campo *Version* establecido en MQCNO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión necesario.

Juego de caracteres y codificación

Los datos de MQCNO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 479. Campos en MQCNO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQCNO_STRUC_ID	'CNO¬'
Versión (número de versión de estructura)	MQCNO_VERSION_1	1
Options (opciones que controlan la acción de MQCONNX)	MQCNO_NONE	0
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_2.		
ClientConnDesplazamiento (desplazamiento de la estructura MQCD para la conexión de cliente)	Ninguna	0

Tabla 479. Campos en MQCNO (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
ClientConnPtr (dirección de la estructura MQCD para la conexión de cliente)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_3.		
ConnTag (etiqueta de conexión del gestor de colas)	MQCT_NONE	Nulos
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_4.		
SSLConfigPtr (dirección de la estructura MQSCO para la conexión de cliente)	Ninguna	Puntero nulo o bytes nulos
SSLConfigOffset (desplazamiento de la estructura MQSCO para la conexión de cliente)	Ninguna	0
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_5.		
ConnectionId (ID de conexión exclusivo)	Ninguna	Puntero nulo o bytes nulos
SecurityParmsDesplazamiento (desplazamiento de la estructura MQSCO para parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
SecurityParmsPtr (dirección de la estructura MQSCO para parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_6.		
Reservado (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.
CCDTUrlLength (longitud de URL de CCDT)	Ninguna	Longitud de la serie identificada por *CCDTUrlPtr* o *CCDTUrlOffset*
CCDTUrlPtr (puntero de URL de CCDT)	Ninguna	Puntero a una serie que contiene un URL, para identificar la ubicación de la tabla de canales de conexión de cliente que se va a utilizar para la conexión.
CCDTUrlOffset (desplazamiento de URL de CCDT)	Ninguna	Desplazamiento en bytes de una serie que contiene un URL que identifica la ubicación de la tabla de canales de conexión de cliente que se va a utilizar para la conexión.

Tabla 479. Campos en MQCNO (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
V 9.1.2 V 9.1.2 **Nota:** Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_7.		
V 9.1.2 ApplName (nombre establecido por la aplicación)	Ninguna	Nombre establecido por la aplicación para identificar la conexión con el gestor de colas
V 9.1.2 Reserved2 (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCNO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQCNO

LTS

```
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         /* Pointer to string containing URL */
  MQBYTE4    Reserved           /* Reserved field to pad out to 64 bit boundary */
  };
```

V 9.1.2

```
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         /* Pointer to 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 */
};
```

Declaración COBOL para MQCNO

LTS

```
**   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 CCDTUrlPtr or CCDTUrlOffset
   15 MQCNO-CCDTURLLENGTH
**    Pointer to a string which contains a URL, to identify the location of the client
connection channel
   15 MQCNO-CCDTURLPTR
**    Offset in bytes from a 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
```

V 9.1.2

```
**   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 CCDTUrlPtr or CCDTUrlOffset
   15 MQCNO-CCDTURLLENGTH
**    Pointer to a string which contains a URL, to identify the location of the client
connection channel
   15 MQCNO-CCDTURLPTR
**    Offset in bytes from a 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
```

Declaración PL/I para MQCNO

LTS

```
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 */
```

V 9.1.2

```
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 */
```

Declaración de High Level Assembler para MQCNO

```
LTS
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
```

```
V 9.1.2
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
```

Declaración de Visual Basic para MQCNO

```
LTS
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'
  End Type
```

V 9.1.2

```
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'
  End Type
```

Tareas relacionadas

Utilización de MQCONNX

StrucId (MQCHAR4)

StrucId es siempre un campo de entrada. Su valor inicial es MQCNO_STRUC_ID.

El valor debe ser:

MQCNO_STRUC_ID
Identificador de la estructura de opciones de conexión.

Para el lenguaje de programación C, también se define la constante MQCNO_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQCNO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)

La versión es siempre un campo de entrada. Su valor inicial es MQCNO_VERSION_1.

El valor debe ser uno de los siguientes:

MQCNO_VERSION_1
Estructura de opciones de conexión de Version-1 .

MQCNO_VERSION_2
Estructura de opciones de conexión de Version-2 .

MQCNO_VERSION_3
Estructura de opciones de conexión Version-3 .

MQCNO_VERSION_4
Estructura de opciones de conexión de Version-4 .

MQCNO_VERSION_5
Estructura de opciones de conexión Version-5 .

MQCNO_VERSION_6
Estructura de opciones de conexión de Version-6 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCNO_CURRENT_VERSION
Versión actual de la estructura de opciones de conexión.

Opciones (MQLONG)

Opciones que controlan la acción de MQCONNX.

Opciones de contabilidad

Las opciones siguientes controlan el tipo de contabilidad si el atributo de gestor de colas **AccountingConnOverride** se establece en MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED
Cuando la recopilación de datos de supervisión está inhabilitada en la definición del gestor de colas estableciendo el atributo **MQIAccounting** en MQMON_OFF, el establecimiento de este distintivo habilita la recopilación de datos de contabilidad MQI.

MQCNO_ACCOUNTING_MQI_DISABLED
Cuando la recopilación de datos de supervisión está inhabilitada en la definición del gestor de colas estableciendo el atributo **MQIAccounting** en MQMON_OFF, el establecimiento de este distintivo detiene la recopilación de datos de contabilidad de MQI.

MQCNO_ACCOUNTING_Q_ENABLED
Cuando la recopilación de datos de contabilidad de colas está inhabilitada en la definición del gestor de colas estableciendo el atributo **MQIAccounting** en MQMON_OFF, el establecimiento de este distintivo permite la recopilación de datos de contabilidad para las colas que especifican un gestor de colas en el campo *MQIAccounting* de su definición de cola.

MQCNO_ACCOUNTING_Q_DISABLED
Cuando la recopilación de datos de contabilidad de colas se inhabilita en la definición del gestor de colas estableciendo el atributo **MQIAccounting** en MQMON_OFF, el establecimiento de este distintivo desactiva la recopilación de datos de contabilidad para las colas que especifican un gestor de colas en el campo *MQIAccounting* de su definición de cola.

Si no se ha definido ninguno de estos distintivos, la contabilidad de la conexión es la que se ha definido en los atributos del gestor de colas.

Opciones de enlace

Las opciones siguientes controlan el tipo de enlace IBM MQ que se va a utilizar. Especifique sólo una de estas opciones:

MQCNO_STANDARD_BINDING
La aplicación y el agente del gestor de colas local (el componente que gestiona las operaciones de puesta en cola) se ejecutan en unidades de ejecución separadas (normalmente, en procesos separados). Esta disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas errantes.

Si el gestor de colas da soporte a varios tipos de enlace y establece MQCNO_STANDARD_BINDING, el gestor de colas utiliza el atributo **DefaultBindType** en la stanza `Connection` del archivo `qm.ini` para seleccionar el tipo real de enlace. Si esta stanza no está definida, o el valor no se puede utilizar o

no es adecuado para la aplicación, el gestor de colas selecciona un tipo de enlace adecuado. El gestor de colas establece el tipo de enlace real utilizado en las opciones de conexión.

Utilice MQCNO_STANDARD_BINDING en situaciones en las que es posible que la aplicación no se haya probado completamente o que no sea fiable o no sea fiable. MQCNO_STANDARD_BINDING es el valor predeterminado.

Esta opción está soportada en todos los entornos.

Si se va a enlazar a la biblioteca `mqm`, primero se intentará establecer una conexión de servidor estándar mediante el tipo de enlace predeterminado. Si no se ha podido cargar la biblioteca de servidor subyacente, en su lugar se intenta una conexión de cliente.

- Para cambiar el comportamiento de MQCONN (o MQCONNX si se especifica MQCNO_STANDARD_BINDING), establezca la variable de entorno MQ_CONNECT_TYPE en una de las opciones siguientes. Tenga en cuenta que hay una excepción a esto: si se especifica MQCNO_FASTPATH_BINDING con MQ_CONNECT_TYPE establecido en `LOCAL` o `STANDARD`, el administrador puede degradar las conexiones de vía de acceso rápida sin un cambio relacionado en la aplicación.

Tabla 480. Valores para MQ_CONNECT_TYPE que cambian el comportamiento de MQCONN o MQCONNX

Valor	**Significado**
CLIENT	Sólo se intenta una conexión de cliente.
FASTPATH	Este valor estaba soportado en los releases anteriores, pero ahora se pasa por alto si se ha especificado.
LOCAL	Sólo se intenta una conexión con el servidor. Las conexiones fastpath se reducen a una conexión con el servidor estándar.
STANDARD	Soportado por motivos de compatibilidad con releases anteriores. Este valor se trata ahora como LOCAL.

- Si la variable de entorno MQ_CONNECT_TYPE no se establece cuando se llama a MQCONNX, se intenta una conexión de servidor estándar utilizando el tipo de enlace predeterminado. Si no se puede cargar la biblioteca del servidor, se intenta una conexión de cliente.

MQCNO_FASTPATH_BINDING

La aplicación y el agente del gestor de colas local forman parte de la misma unidad de ejecución. Esto contrasta con el método típico de enlace, donde la aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes.

MQCNO_FASTPATH_BINDING se ignora si el gestor de colas no da soporte a este tipo de enlace; el proceso continúa como si no se hubiera especificado la opción.

MQCNO_FASTPATH_BINDING puede ser una ventaja en situaciones en las que varios procesos consumen más recursos que el recurso global utilizado por la aplicación.Una aplicación que utiliza el enlace de vía de acceso rápida se conoce como *aplicación de confianza*.

Tenga en cuenta los siguientes puntos importantes al decidir si se debe utilizar el enlace de vía de acceso rápida:

- El uso de la opción MQCNO_FASTPATH_BINDING no impide que una aplicación altere o dañe mensajes y otras áreas de datos pertenecientes al gestor de colas. Utilice esta opción sólo en situaciones en las que haya evaluado completamente estos problemas.
- La aplicación no debe utilizar señales asíncronas o interrupciones de temporizador (como `sigkill`) con MQCNO_FASTPATH_BINDING. También hay restricciones sobre el uso de segmentos de memoria compartida.
- La aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.

- La aplicación debe finalizar antes de finalizar el gestor de colas con el mandato `endmqm` .
- IBM i En IBM i, el trabajo debe ejecutarse bajo un perfil de usuario que pertenezca al grupo `QMQMADM` . Además, el programa no debe detenerse de forma anómala, de lo contrario se pueden producir resultados imprevisibles.
- UNIX En UNIX, el identificador de usuario `mqm` debe ser el identificador de usuario efectivo y el identificador de grupo `mqm` debe ser el identificador de grupo efectivo. Para que la aplicación se ejecute de este modo, configure el programa para que sea propiedad del identificador de usuario `mqm` y del identificador de grupo `mqm` y, a continuación, establezca los bits de permiso `setuid` y `setgid` en el programa.

  El Gestor de autorizaciones sobre objetos (OAM) de IBM MQ sigue utilizando el ID de usuario real para la comprobación de autorización.
- Windows En Windows, el programa debe ser miembro del grupo `mqm` . El enlace de vía de acceso rápida no está soportado para aplicaciones de 64 bits.

La opción MQCNO_FASTPATH_BINDING está soportada en los entornos siguientes:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Solaris Solaris
- Windows Windows

z/OS En z/OS, la opción se acepta pero se ignora.

Para obtener más información sobre las implicaciones de utilizar aplicaciones de confianza, consulte Restricciones para aplicaciones de confianza.

MQCNO_SHARED_BINDING

Con MQCNO_SHARED_BINDING, la aplicación y el agente del gestor de colas local comparten algunos recursos. MQCNO_SHARED_BINDING se omite si el gestor de colas no soporta este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_ISOLATED_BINDING

En este caso, el proceso de aplicaciones y el agente del gestor de colas local se aíslan entre sí en el sentido de que no comparten recursos. Si el gestor de colas no da soporte a este tipo de enlace, se ignora MQCNO_ISOLATED_BINDING. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_CLIENT_BINDING

Especifique esta opción para que la aplicación intente únicamente una conexión con el cliente. Esta opción tiene las siguientes limitaciones:

- z/OS En z/OS se omite MQCNO_CLIENT_BINDING.
- MQCNO_CLIENT_BINDING se rechaza con MQRC_OPTIONS_ERROR, si se ha especificado con cualquier opción de enlace MQCNO distinta de MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.

MQCNO_LOCAL_BINDING

Especifique esta opción para hacer que la aplicación intente una conexión con el servidor.
Si también se ha especificado MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING o MQCNO_SHARED_BINDING, la conexión será de dicho tipo, y se ha documentado en esta sección.

De lo contrario, se intenta efectuar una conexión de servidor estándar utilizando el tipo de enlace predeterminado. MQCNO_LOCAL_BINDING tiene las limitaciones siguientes:

- z/OS Se omite MQCNO_LOCAL_BINDING en z/OS.
- MQCNO_LOCAL_BINDING se rechaza con MQRC_OPTIONS_ERROR si se ha especificado con cualquier opción de reconexión MQCNO que no sea MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.

En las plataformas siguientes, puede utilizar la variable de entorno `MQ_CONNECT_TYPE` con el tipo de enlace especificado por el campo `Options` , para controlar el tipo de enlace utilizado.

- AIX AIX
- Linux Linux
- Solaris Solaris
- Windows Windows

Si especifica esta variable de entorno, debe tener el valor `FASTPATH` o `STANDARD` ; si tiene un valor diferente, se ignora. El valor de la variable de entorno distingue entre mayúsculas y minúsculas; consulte Variable de entorno MQCONNX para obtener más información.

La variable de entorno y el campo *`Options`* interactúan de la forma siguiente:

- Si omite la variable de entorno, o le da un valor que no está soportado, el uso del enlace de vía de acceso rápida lo determina únicamente el campo `Options` .
- Si proporciona a la variable de entorno un valor soportado, el enlace de vía de acceso rápida sólo se utiliza si tanto la variable de entorno como el campo `Options` especifican el enlace de vía de acceso rápida.

Opciones de etiqueta de conexión

LTS Estas opciones sólo están soportadas al conectarse a un gestor de colas de z/OS y controlan el uso de la etiqueta de conexión `ConnTag`. Sólo puede especificar una de estas opciones.

V 9.1.3 La implementación precisa de los códigos de conexión difiere entre IBM MQ for z/OS y IBM MQ for Multiplatforms:

- z/OS Las opciones siguientes, aparte de *MQCNO_GENERATE_CONN_TAG*, sólo están soportadas al conectarse a un gestor de colas z/OS y controlan el uso del código de conexión. Sólo puede especificar una de las opciones soportadas.
- ULW *MQCNO_GENERATE_CONN_TAG* sólo está soportado en plataformas distintas de z/OS.

ULW V 9.1.3 **MQCNO_GENERATE_CONN_TAG**

Devuelve el código de conexión que el gestor de colas ha asociado con esta conexión, en la estructura MQCNO de salida.

La etiqueta de conexión devuelta será idéntica para todas las conexiones que el gestor de colas considere como una única instancia de aplicación.

z/OS **MQCNO_SERIALIZE_CONN_TAG_Q_MGR**

Esta opción solicita el uso exclusivo del código de conexión dentro del gestor de colas local. Si el código de conexión ya está en uso en el gestor de colas local, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso de la

etiqueta de conexión en otro lugar del grupo de compartición de colas al que pertenece el gestor de colas local.

z/OS **MQCNO_SERIALIZE_CONN_TAG_QSG**

Esta opción solicita el uso exclusivo del código de conexión dentro del grupo de compartición de colas al que pertenece el gestor de colas local. Si el código de conexión ya está en uso en el grupo de compartición de colas, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE.

z/OS **MQCNO_RESTRICT_CONN_TAG_Q_MGR**

Esta opción solicita el uso compartido del código de conexión dentro del gestor de colas local. Si el código de conexión ya está en uso en el gestor de colas local, la llamada MQCONNX puede tener éxito si la aplicación solicitante se ejecuta en el mismo ámbito de proceso que el usuario existente del código. Si no se cumple esta condición, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso del código de conexión en otro lugar del grupo de compartición de colas al que pertenece el gestor de colas local.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_Q_MGR.

z/OS **MQCNO_RESTRICT_CONN_TAG_QSG**

Esta opción solicita el uso compartido del código de conexión dentro del grupo de compartición de colas al que pertenece el gestor de colas local. Si el código de conexión ya está en uso en el grupo de compartición de colas, la llamada MQCONNX puede ser satisfactoria siempre que la aplicación solicitante se esté ejecutando en el mismo ámbito de proceso y esté conectada al mismo gestor de colas, que el usuario existente del código.

Si no se cumplen estas condiciones, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_QSG.

Si no se especifica ninguna de estas opciones, no se utiliza `ConnTag` . Estas opciones no son válidas si `Version` es menor que MQCNO_VERSION_3.

Opciones de manejo compartido

Multi

Estas opciones están soportadas en los entornos siguientes:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Solaris Solaris
- Windows Windows

Controlan la compartición de manejadores entre diferentes hebras (unidades de proceso paralelo) dentro del mismo proceso. Sólo puede especificar una de estas opciones:

MQCNO_HANDLE_SHARE_NONE

Esta opción indica que la conexión y los descriptores de contexto de objeto sólo pueden ser utilizados por la hebra que ha hecho que se asignara el descriptor de contexto (es decir, la hebra que ha emitido

la llamada MQCONN, MQCONNX o MQOPEN). Otras hebras que pertenecen al mismo proceso no pueden utilizar los descriptores de contexto.

MQCNO_HANDLE_SHARE_BLOCK

Esta opción indica que las conexiones y los manejadores de objetos asignados por una hebra de un proceso pueden ser utilizados por otras hebras pertenecientes al mismo proceso. Sin embargo, sólo una hebra a la vez puede utilizar un descriptor de contexto determinado; es decir, sólo se permite el uso en serie de un descriptor de contexto. Si una hebra intenta utilizar un descriptor de contexto que ya está en uso por otra hebra, la llamada se bloquea (espera) hasta que el descriptor de contexto esté disponible.

MQCNO_HANDLE_SHARE_NO_BLOCK

Es lo mismo que MQCNO_HANDLE_SHARE_BLOCK, excepto que si el descriptor de contexto está siendo utilizado por otra hebra, la llamada se completa inmediatamente con MQCC_FAILED y MQRC_CALL_IN_PROGRESS en lugar de bloquearse hasta que el descriptor de contexto esté disponible.

Una hebra puede tener cero o un descriptor de contexto no compartido:

- Cada llamada MQCONN o MQCONNX que especifica MQCNO_HANDLE_SHARE_NONE devuelve un nuevo descriptor de contexto no compartido en la primera llamada, y el mismo descriptor de contexto no compartido en la segunda y posteriores llamadas (suponiendo que no haya ninguna llamada MQDISC interviniente). El código de razón es MQRC_ALREADY_CONNECTED para la segunda llamada y posteriores.
- Cada llamada MQCONNX que especifica MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK devuelve un nuevo descriptor de contexto compartido en cada llamada.

Los descriptores de contexto de objeto heredan las mismas propiedades de compartición que el descriptor de contexto de conexión especificado en la llamada MQOPEN que ha creado el descriptor de contexto de objeto. Además, las unidades de trabajo heredan las mismas propiedades de compartición que el descriptor de conexión utilizado para iniciar la unidad de trabajo; si la unidad de trabajo se inicia en una hebra utilizando un descriptor de contexto compartido, la unidad de trabajo se puede actualizar en otra hebra utilizando el mismo descriptor de contexto.

Si no especifica una opción de uso compartido de descriptor de contexto, el valor predeterminado lo determina el entorno:

- Windows En el entorno de Microsoft Transaction Server (MTS), el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_BLOCK.
- En otros entornos, el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_NONE.

Opciones de reconexión

Las opciones de reconexión determinan si una conexión es reconectable. Sólo se pueden volver a conectar las conexiones de cliente.

MQCNO_RECONNECT_AS_DEF

La opción de reconexión se resuelve en su valor predeterminado. Si no se establece ningún valor predeterminado, el valor de esta opción se resuelve en `DISABLED`. El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

MQCNO_RECONNECT

La aplicación se puede reconectar a cualquier gestor de colas coherente con el valor del parámetro **`QmgrName`** de `MQCONNX`. Utilice la opción MQCNO_RECONNECT sólo si no hay afinidad entre la aplicación cliente y el gestor de colas con el que estableció inicialmente una conexión. El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

MQCNO_RECONNECT_DISABLED

La aplicación no se puede volver a conectar. El valor de la opción no se pasa al servidor.

MQCNO_RECONNECT_Q_MGR

La aplicación sólo se puede volver a conectar al gestor de colas con el que se conectó originalmente. Utilice este valor si un cliente se puede reconectar, pero existe una afinidad entre la aplicación cliente y el gestor de colas con el que estableció originalmente una conexión. Elija este valor si desea que el cliente se reconecte automáticamente a la instancia en espera de un gestor de colas con elevada disponibilidad. El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

Utilice las opciones MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED y MQCNO_RECONNECT_Q_MGR sólo para conexiones de cliente. Si las opciones se utilizan para una conexión de enlace, `MQCONNX` falla con el código de terminación `MQCC_FAILED` y el código de razón `MQRC_OPTIONS_ERROR`. La reconexión automática de cliente no está soportada por IBM MQ classes for Java

Opciones de compartición de conversación

Las opciones siguientes sólo se aplican a las conexiones de cliente TCP/IP. Para los canales SNA, SPX y NetBios , estos valores se ignoran y el canal se ejecuta como en las versiones anteriores del producto

MQCNO_NO_CONV_SHARING

Esta opción no permite el uso compartido de conversación.

Puede utilizar MQCNO_NO_CONV_SHARING en situaciones en las que las conversaciones están muy cargadas y, por lo tanto, donde la contención es una posibilidad en el extremo de conexión de servidor de la instancia de canal en la que existen las conversaciones de compartición. MQCNO_NO_CNV_SHARING se comporta como sharecnv (1) cuando se conecta a un canal que da soporte a la compartición de conversación, y sharecnv (0) cuando se conecta a un canal que no da soporte a la compartición de conversación.

MQCNO_ALL_CONVS_SHARE

Esta opción permite el uso compartido de conversaciones; la aplicación no pone ningún límite en el número de conexiones en la instancia de canal. Esta opción es el valor predeterminado.

Si la aplicación indica que la instancia de canal puede compartir, pero la definición de *`SharingConversations`* (SHARECNV) en el extremo de conexión de servidor del canal está establecida en uno, no se produce ninguna compartición y no se proporciona ningún aviso a la aplicación.

De forma similar, si la aplicación indica que la compartición está permitida pero la definición de *`SharingConversations`* de conexión con el servidor está establecida en cero, no se proporciona ningún aviso y la aplicación muestra el mismo comportamiento que un cliente en versiones del producto anteriores a IBM WebSphere MQ 7.0; se ignora el valor de la aplicación relacionado con la compartición de conversaciones.

MQCNO_NO_CONV_SHARING y MQCNO_ALL_CONVS_SHARE se excluyen mutuamente. Si se especifican ambas opciones en una conexión determinada, la conexión se rechaza con un código de razón de MQRC_OPTIONS_ERROR.

Opciones de definición de canal

Las opciones siguientes controlan el uso de la estructura de definición de canal pasada en MQCNO:

MQCNO_CD_FOR_OUTPUT_ONLY

Esta opción permite que la estructura de definición de canal en MQCNO sólo se utilice para devolver el nombre de canal utilizado en una llamada MQCONNX satisfactoria.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de razón MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como un cliente, la opción se ignora.

El nombre de canal devuelto puede utilizarse en una llamada MQCONNX posterior utilizando la opción MQCNO_USE_CD_SELECTION para volver a conectarse utilizando la misma definición de canal. Esto puede ser útil cuando hay varias definiciones de canal aplicables en la tabla de canales de cliente.

MQCNO_USE_CD_SELECTION

Esta opción permite que la llamada MQCONNX se conecte utilizando el nombre de canal contenido en la estructura de definición de canal pasada en MQCNO.

Si se establece la variable de entorno MQSERVER, se utiliza la definición de canal definida por la misma. Si no se establece MQSERVER, se utiliza la tabla de canales de cliente.

Si no se encuentra una definición de canal con un nombre de canal y un nombre de gestor de colas coincidentes, la llamada falla con el código de razón MQRC_Q_MGR_NAME_ERROR.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de razón MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como un cliente, la opción se ignora.

Opción predeterminada

Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

MQCNO_NONE

No se especifica ninguna opción.

Utilice MQCNO_NONE para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción MQCNO_ *, pero debido a que su valor es cero, no se puede detectar dicho uso.

ClientConnDesplazamiento (MQLONG)

ClientConnDesplazamiento es el desplazamiento en bytes de una estructura de definición de canal MQCD desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada con un valor inicial de 0.

Utilice *ClientConnOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client. Para obtener información sobre cómo utilizar este campo, consulte la descripción del campo *ClientConnPtr* .

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

ClientConnPtr (MQPTR)

ClientConnPtr es un campo de entrada. Su valor inicial es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Utilice *ClientConnOffset* y *ClientConnPtr* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como IBM MQ MQI client. Al especificar uno u otro de estos campos, la aplicación puede controlar la definición del canal de conexión de cliente proporcionando una estructura de definición de canal MQCD que contenga los valores necesarios.

Si la aplicación se ejecuta como un IBM MQ MQI client, pero no proporciona una estructura MQCD, se utiliza la variable de entorno MQSERVER para seleccionar la definición de canal. Si no se establece MQSERVER , se utiliza la tabla de canales de cliente.

Si la aplicación no se ejecuta como IBM MQ MQI client, se ignoran *ClientConnOffset* y *ClientConnPtr* .

Si la aplicación proporciona una estructura MQCD, establezca los campos listados en los valores necesarios; los demás campos de MQCD se ignoran. Puede rellenar las series de caracteres con espacios en blanco hasta la longitud del campo, o terminarlas con un carácter nulo. Consulte "Campos" en la página 1519 para obtener más información sobre los campos de la estructura MQCD.

Tabla 481. Campos en MQCD

Campo en MQCD	Valor
ChannelName	Nombre de canal.
Version	Número de versión de la estructura. No debe ser menor que MQCD_VERSION_7.
TransportType	Cualquier tipo de transporte soportado.
ModeName	Nombre de modalidad de LU 6.2 .
TpName	Nombre de programa de transacción de LU 6.2 .
SecurityExit	Nombre de la salida de seguridad de canal.
SendExit	Nombre de la salida de envío de canal.
ReceiveExit	Nombre de la salida de recepción de canal.
MaxMsgLength	Longitud máxima en bytes de mensajes que se pueden enviar a través del canal de conexión de cliente.
SecurityUserData	Datos de usuario para salida de seguridad.
SendUserData	Datos de usuario para salida de envío.
ReceiveUserData	Datos de usuario para salida de recepción.
UserIdentifier	Identificador de usuario que se utilizará para establecer una sesión de LU 6.2 .
Password	Contraseña que se utilizará para establecer una sesión de LU 6.2 .
ConnectionName	Nombre de conexión.
HeartbeatInterval	Tiempo en segundos entre flujos de pulsaciones.
StrucLength	Longitud de la estructura MQCD.
ExitNameLength	Longitud de los nombres de salida a los que se dirigen *SendExitPtr* y *ReceiveExitPtr*. Debe ser mayor que cero si *SendExitPtr* o *ReceiveExitPtr* se establece en un valor que no sea el puntero nulo.
ExitDataLength	Longitud de los datos de salida direccionado por *SendUserDataPtr* y *ReceiveUserDataPtr*. Debe ser mayor que cero si *SendUserDataPtr* o *ReceiveUserDataPtr* se establece en un valor que no sea el puntero nulo.
SendExitsDefined	Número de salidas de envío dirigidas por *SendExitPtr*. Si es cero, *SendExit* y *SendUserData* proporcionan el nombre de salida y los datos. Si es mayor que cero, *SendExitPtr* y *SendUserDataPtr* proporcionan los nombres y datos de salida, y *SendExit* y *SendUserData* deben estar en blanco.
ReceiveExitsDefined	Número de salidas de recepción dirigidas por *ReceiveExitPtr*. Si es cero, *ReceiveExit* y *ReceiveUserData* proporcionan el nombre de salida y los datos. Si es mayor que cero, *ReceiveExitPtr* y *ReceiveUserDataPtr* proporcionan los nombres y datos de salida, y *ReceiveExit* y *ReceiveUserData* deben estar en blanco.
SendExitPtr	Dirección del nombre de la primera salida de envío.

Tabla 481. Campos en MQCD (continuación)

Campo en MQCD	Valor
SendUserDataPtr	Dirección de datos para la primera salida de envío.
ReceiveExitPtr	Dirección del nombre de la primera salida de recepción.
ReceiveUserDataPtr	Dirección de datos para la primera salida de recepción.
LongRemoteUserIdLength	Longitud del identificador de usuario remoto largo.
LongRemoteUserIdPtr	Dirección del identificador de usuario remoto largo.
RemoteSecurityId	Identificador de seguridad remota.
SSLCipherSpec	CipherSpecde TLS.
SSLPeerNamePtr	Dirección del nombre de igual TLS.
SSLPeerNameLength	Longitud del nombre de igual TLS.
KeepAliveInterval	Valor pasado a la pila de comunicaciones para la temporización de estado activo para el canal
LocalAddress	La dirección de comunicaciones local, incluida la dirección IP del adaptador de red local que se va a utilizar, y un rango de puertos que se van a utilizar para las conexiones de salida.

Proporcione la estructura de definición de canal de una de estas dos maneras:

- Utilizando el campo de desplazamiento *ClientConnOffset*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga un MQCNO seguido de la estructura de definición de canal MQCD y establecer *ClientConnOffset* en el desplazamiento de la estructura de definición de canal desde el inicio del MQCNO. Asegúrese de que este desplazamiento sea correcto. *ClientConnPtr* debe establecerse en el puntero nulo o en bytes nulos.

  Utilice *ClientConnOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

  Para el lenguaje de programación Visual Basic, una estructura compuesta denominadaMQCNOCD se proporciona en el archivo de cabecera CMQXB.BAS; esta estructura contiene una estructura MQCNO seguida de una estructura MQCD. Inicialice MQCNOCD invocando la subrutina MQCNOCD_DEFAULTS. MQCNOCD se utiliza con elVariante MQCONNXAny de la llamada MQCONNX; consulte la descripción de la llamada MQCONNX para obtener más detalles.
- Utilizando el campo de puntero *ClientConnPtr*

  En este caso, la aplicación puede declarar la estructura de definición de canal por separado de la estructura MQCNO y establecer *ClientConnPtr* en la dirección de la estructura de definición de canal. Establezca *ClientConnOffset* en cero.

  Utilice *ClientConnPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

  En el lenguaje de programación C, puede utilizar la variable de macro MQCD_CLIENT_CONN_DEFAULT para proporcionar valores iniciales para la estructura que son más adecuados para su uso en la llamada MQCONNX que los valores iniciales proporcionados por MQCD_DEFAULT.

Sea cual sea la técnica que elija, solo puede utilizar uno de *ClientConnOffset* y *ClientConnPtr* ; la llamada falla con el código de razón MQRC_CLIENT_CONN_ERROR si ambos son distintos de cero.

Cuando se ha completado la llamada MQCONNX, no se vuelve a hacer referencia a la estructura MQCD.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

V 9.1.3 *ConnTag (MQBYTE128) en Multiplatforms*

Una etiqueta de conexión es conceptualmente similar a un identificador de conexión, pero puede abarcar varias conexiones relacionadas, identificándolas como una única instancia de aplicación. En Multiplatforms, el gestor de colas genera la etiqueta de conexión en el momento de la conexión.

V 9.1.3 Para obtener más información, consulte identificador de conexión y instancia de aplicación.

Las etiquetas de conexión generadas son `semi` legibles por el usuario. Es decir, se pueden visualizar y filtrar en MQSC como si fueran series en el juego de caracteres local. A las conexiones que IBM MQ sabe que están relacionadas se les asigna automáticamente la misma etiqueta de conexión. Esta asignación es especialmente importante para el equilibrio de aplicaciones.

La etiqueta de conexión generada es visible de tres maneras:

- En la estructura MQCNO de salida en una llamada MQCONNX, cuando se especifica MQCNO_GENERATE_CONN_TAG .
- En la salida de DISPLAY CONN (o equivalentes programáticos).
- En la salida de DISPLAY APSTATUS (o equivalentes).

La etiqueta deja de ser válida cuando la aplicación termina o emite la llamada MQDISC.

Referencia relacionada

"ConnTag (MQBYTE128) en IBM MQ for z/OS" en la página 333
Una etiqueta de conexión es conceptualmente similar a un identificador de conexión, pero puede abarcar varias conexiones relacionadas, identificándolas como una única instancia de aplicación. En IBM MQ for z/OS, el código de conexión es un campo de entrada, proporcionado por la aplicación y utilizado junto con las opciones MQCNO_ * _CONN_TAG para serializar conexiones desde esa instancia de aplicación.

z/OS *ConnTag (MQBYTE128) en IBM MQ for z/OS*

Una etiqueta de conexión es conceptualmente similar a un identificador de conexión, pero puede abarcar varias conexiones relacionadas, identificándolas como una única instancia de aplicación. En IBM MQ for z/OS, el código de conexión es un campo de entrada, proporcionado por la aplicación y utilizado junto con las opciones MQCNO_ * _CONN_TAG para serializar conexiones desde esa instancia de aplicación.

Cuando hay varias instancias de una aplicación que están pensadas para conectarse simultáneamente, cada una de ellas debe proporcionar un valor exclusivo para este campo. Consulte las descripciones de estas opciones de etiqueta de conexión para obtener más detalles.

Notas:

- En IBM MQ for z/OS, no hay forma de determinar administrativamente la etiqueta de conexión asociada con una aplicación en tiempo de ejecución.
- Los valores de etiqueta de conexión que empiezan por `MQ` en mayúsculas, minúsculas o mayúsculas y minúsculas en ASCII o EBCDIC están reservados para que los utilicen los productos IBM . No utilice valores de etiqueta de conexión que empiecen por estas letras.

Utilice el siguiente valor especial si no necesita ninguna etiqueta:

MQCT_NONE
El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCT_NONE_ARRAY; esta constante tiene el mismo valor que MQCT_NONE, pero es una matriz de caracteres en lugar de una serie.

El campo ConnTag se utiliza al conectarse a un gestor de colas de z/OS .

La longitud de este campo la proporciona MQ_CONN_TAG_LENGTH. Este campo se ignora si *Version* es menor que MQCNO_VERSION_3.

Multi Consulte “ConnTag (MQBYTE128) en Multiplatforms” en la página 333 para obtener información sobre cómo utilizar la etiqueta de conexión en IBM MQ for Multiplatforms.

SSLConfigPtr (PMQSCO)

SSLConfigPtr es un campo de entrada. Su valor inicial es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Utilice *SSLConfigPtr* y *SSLConfigOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client y el protocolo de canal sea TCP/IP. Si la aplicación no se ejecuta como un cliente IBM MQ , o el protocolo de canal no es TCP/IP, se ignoran *SSLConfigPtr* y *SSLConfigOffset* .

Al especificar *SSLConfigPtr* o *SSLConfigOffset*, más *ClientConnPtr* o *ClientConnOffset*, la aplicación puede controlar el uso de TLS para la conexión de cliente. Cuando la información de TLS se especifica de esta forma, las variables de entorno MQSSLKEYR y MQSSLCRYP se ignoran; cualquier información relacionada con TLS en la tabla de definiciones de canal de cliente (CCDT) también se ignora.

La información de TLS solo se puede especificar en:

- La primera llamada MQCONNX del proceso de cliente, o
- Una llamada MQCONNX posterior cuando todas las conexiones TLS anteriores con el gestor de colas han finalizado utilizando MQDISC.

Estos son los únicos estados en los que se puede inicializar el entorno TLS de todo el proceso. Si se emite una llamada MQCONNX especificando información TLS cuando el entorno TLS ya existe, la información TLS de la llamada se ignora y la conexión se realiza utilizando el entorno TLS existente; la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SSL_ALREADY_INITIALIZED en este caso.

Puede proporcionar la estructura MQSCO de la misma forma que la estructura MQCD, especificando una dirección en *SSLConfigPtr*o especificando un desplazamiento en *SSLConfigOffset* ; consulte la descripción de *ClientConnPtr* para obtener detalles sobre cómo hacerlo. Sin embargo, no puede utilizar más de uno de *SSLConfigPtr* y *SSLConfigOffset* ; La llamada falla con el código de razón MQRC_SSL_CONFIG_ERROR. si ambos son distintos de cero.

Una vez completada la llamada MQCONNX, no se vuelve a hacer referencia a la estructura MQSCO.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

SSLConfigOffset (MQLONG)

SSLConfigOffset es el desplazamiento en bytes de una estructura MQSCO desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Utilice *SSLConfigOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client. Para obtener información sobre cómo utilizar este campo, consulte la descripción del campo *SSLConfigPtr* .

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

ConnectionId (MQBYTE24)

ConnectionId es un identificador exclusivo de 24 bytes que permite a IBM MQ identificar de forma fiable una aplicación. Una aplicación puede utilizar este identificador para la correlación en llamadas PUT y GET. Este parámetro de salida tiene un valor inicial de 24 bytes nulos en todos los lenguajes de programación.

El gestor de colas asigna un ID exclusivo a todas las conexiones, sin embargo, se establecen. Si un MQCONNX establece la conexión con un MQCNO de la versión 5, la aplicación puede determinar el ConnectionId del MQCNO devuelto. Se garantiza que el identificador asignado es exclusivo entre todos los demás identificadores que genera IBM MQ , como por ejemplo CorrelId, MsgIDy GroupId.

Utilice ConnectionId para identificar unidades de trabajo de larga ejecución utilizando el mandato PCF Inquire Connection o el mandato MQSC DISPLAY CONN. El ConnectionId utilizado por los mandatos MQSC (CONN) se deriva del ConnectionId devuelto aquí. Los mandatos Consultar y Detener conexión de PCF pueden utilizar el ConnectionId devuelto aquí sin modificación.

Puede utilizar el ConnectionId para forzar el final de una unidad de trabajo de larga ejecución, especificando el ConnectionId utilizando el mandato PCF Detener conexión o el mandato MQSC STOP CONN. Consulte Detener conexión y STOP CONN para obtener más información sobre cómo utilizar estos mandatos.

Este campo no se devuelve si la versión es inferior a MQCNO_VERSION_5.

La longitud de este campo la proporciona MQ_CONNECTION_ID_LENGTH.

SecurityParmsDesplazamiento (MQLONG)

SecurityParmsDesplazamiento es el desplazamiento en bytes de la estructura MQCSP desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Este campo se ignora si *Versión* es menor que MQCNO_VERSION_5.

La estructura MQCSP se define en “MQCSP-Parámetros de seguridad” en la página 336.

SecurityParmsPtr (PMQCSP)

SecurityParmsPtr es la dirección de la estructura MQCSP, utilizada para especificar un ID de usuario y una contraseña para la autenticación por parte del servicio de autorización. Este campo es un campo de entrada y su valor inicial es un puntero nulo o bytes nulos.

Este campo se ignora si *Versión* es menor que MQCNO_VERSION_5.

La estructura MQCSP se define en “MQCSP-Parámetros de seguridad” en la página 336.

Reservado (MQBYTE4)

Un campo reservado para rellenar la estructura hasta un límite de 64 bits. El valor inicial del campo es cero binario para la longitud del campo.

Este campo se ignora si `Version` es menor que MQCNO_VERSION_6.

CCDTUrlLength (MQLONG)

CCDTUrlLength es la longitud de la serie identificada por CCDTUrlPtr o CCDTUrlOffset que contiene un URL que identifica la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión. El valor inicial del campo es cero.

Utilice `CCDTUrlLength` sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora `CCDTUrlLength` .

Este campo se ignora si `Version` es menor que MQCNO_VERSION_6.

CCDTUrlPtr (PMQCHAR)

CCDTUrlPtr es un puntero opcional a una serie que contiene un URL, para identificar la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión. Este campo es un campo de entrada, con un valor inicial de un puntero nulo en lenguajes de programación que dan soporte a punteros, y una serie de bytes totalmente nula en caso contrario.

Utilice `CCDTUrlPtr` sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client.

Importante: Sólo puede utilizar uno de `CCDTUrlPtr` y `CCDTUrlOffset`. La llamada falla con el código de razón MQRC_CCDT_URL_ERROR si ambos campos son distintos de cero.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora `CCDTUrlPtr` .

Este campo se ignora si `Version` es menor que MQCNO_VERSION_6.

CCDTUrlOffset (MQLONG)

CCDTUrlOffset es el desplazamiento en bytes, desde el inicio de la estructura MQCNO, a una serie que contiene un URL que identifica la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión. El desplazamiento puede ser positivo o negativo y el valor inicial del campo es cero.

Utilice `CCDTUrlOffset` sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client.

Importante: Sólo puede utilizar uno de `CCDTUrlPtr` y `CCDTUrlOffset`. La llamada falla con el código de razón MQRC_CCDT_URL_ERROR si ambos campos son distintos de cero.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora `CCDTUrlOffset` .

Este campo se ignora si `Version` es menor que MQCNO_VERSION_6.

V 9.1.2 *ApplName (MQCHAR28)*

El nombre establecido por la aplicación para identificar la conexión con el gestor de colas. El valor inicial del campo es MQAN_NONE_ARRAY (caracteres en blanco).

Este campo se ignora si `Version` es menor que MQCNO_VERSION_7, o si el valor se establece en blancos.

z/OS No puede establecer este campo en z/OS. Si intenta hacerlo, recibirá de nuevo el código de razón MQRC_CNO_ERROR.

V 9.1.2 *Reserved2 (MQBYTE4)*

Un campo reservado para rellenar la estructura hasta un límite de 64 bits. El valor inicial del campo es cero binario para la longitud del campo.

Este campo se ignora si `Version` es menor que MQCNO_VERSION_7.

MQCSP-Parámetros de seguridad

La estructura MQCSP permite al servicio de autorización autenticar un ID de usuario y una contraseña. Especifique la estructura de parámetros de seguridad de conexión MQCSP en una llamada MQCONNX.

Aviso: En algunos casos, la contraseña de una estructura MQCSP para una aplicación cliente se envía a través de la red en texto sin formato. Para asegurarse de que las contraseñas de aplicación cliente están protegidas adecuadamente, consulte Protección de contraseñas MQCSP.

Disponibilidad

La estructura MQCSP está disponible en todas las plataformas IBM MQ soportadas.

Juego de caracteres y codificación

Los datos de MQCSP deben estar en el juego de caracteres y la codificación del gestor de colas local, proporcionados por el atributo de gestor de colas **`CodedCharSetId`** y MQENC_NATIVE, respectivamente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 482. Campos en MQCSP

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQCSP_STRUC_ID	'CSP¬'
Versión (número de versión de estructura)	MQCSP_VERSION_1	1
AuthenticationType (tipo de autenticación)	Ninguna	MQCSP_AUTH_NONE
Reserved1 (necesario para la alineación de puntero en IBM i)	Ninguna	Serie nula o espacios en blanco
CSPUserIdPtr (dirección del ID de usuario)	Ninguna	Puntero nulo o bytes nulos
CSPUserIdDesplazamiento (desplazamiento del ID de usuario)	Ninguna	0
CSPUserIdLongitud (longitud del ID de usuario)	Ninguna	0
Reserved2 (necesario para la alineación de puntero en IBM i)	Ninguna	Serie nula o espacios en blanco
CSPPasswordPtr (dirección de contraseña)	Ninguna	Puntero nulo o bytes nulos
CSPPasswordOffset (desplazamiento de contraseña)	Ninguna	0
CSPPasswordLength (longitud de contraseña)	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCSP_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQCSP

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer
                                    alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer
                                    alignment */
                                  alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
```

```
  MQLONG     CSPPasswordLength;  /* Length of password */
  };
```

Declaración COBOL para MQCSP

```
**   MQCSP structure
  10 MQCSP.
**    Structure identifier
   15 MQCSP-STRUCID           PIC X(4).
**    Structure version number
   15 MQCSP-VERSION           PIC S9(9) BINARY.
**    Type of authentication
   15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED1         PIC X(4).
**    Address of user ID
   15 MQCSP-CSPUSERIDPTR      POINTER.
**    Offset of user ID
   15 MQCSP-CSPUSERIDOFFSET   PIC S9(9) BINARY.
**    Length of user ID
   15 MQCSP-CSPUSERIDLENGTH   PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED2         PIC X(4).
**    Address of password
   15 MQCSP-CSPPASSWORDPTR    POINTER.
**    Offset of password
   15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**    Length of password
   15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
```

Declaración PL/I para MQCSP

```
dcl
 1 MQCSP based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 AuthenticationType fixed bin(31),  /* Type of authentication */
  3 Reserved1         char(4),     /* Required for IBM i pointer
                                          alignment */
  3 CSPUserIdPtr      pointer,      /* Address of user ID */
  3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
  3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
  3 Reserved2         char(8),     /* Required for IBM i pointer
                                          alignment */
  3 CSPPasswordPtr    pointer,      /* Address of password */
  3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
  3 CSPPasswordLength fixed bin(31); /* Length of user ID */
```

Declaración de Visual Basic para 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
```

StrucId (MQCHAR4)

Identificador de estructura.

El valor debe ser:

MQCSP_STRUC_ID
Identificador de la estructura de parámetros de seguridad.

Para el lenguaje de programación C, también se define la constante MQCSP_STRUC_ID_ARRAY; tiene el mismo valor que MQCSP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCSPSTRUC_ID.

Versión (MQLONG)
Número de versión de la estructura.

El valor debe ser:

MQCSP_VERSION_1
Estructura de parámetros de seguridad de Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCSP_CURRENT_VERSION
Versión actual de la estructura de parámetros de seguridad.

Siempre es un campo de entrada. El valor inicial de este campo es MQCSP_VERSION_1.

AuthenticationType (MQLONG)
AuthenticationType es un campo de entrada. Su valor inicial es MQCSP_AUTH_NONE.

Este es el tipo de autenticación que se debe realizar. Los valores válidos son:

MQCSP_AUTH_NONE
No utilice los campos de ID de usuario y contraseña.

MQCSP_AUTH_USER_ID_AND_PWD
Autentique los campos de ID de usuario y contraseña.

El valor predeterminado es MQCSP_AUTH_NONE. Con el valor predeterminado, no se lleva a cabo ninguna protección por contraseña.

Si necesita autenticación, debe establecer **MQCSP**.**AuthenticationType** a MQCSP_AUTH_USER_ID_AND_PWD.

Consulte Protección de contraseña MQCSP para obtener más información.

Reserved1 (MQBYTE4)
Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSPUserIdPtr (MQPTR)
Es la dirección en bytes del ID de usuario que se va a utilizar en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

Este campo puede contener un ID de usuario del sistema operativo cuando un **AUTHTYPE** de *IDPWOS* se denomina en el campo CONNAUTH del gestor de colas.

En Windows , puede ser un ID de usuario de dominio completo.

Este campo puede contener un ID de usuario LDAP cuando un **AUTHTYPE** de *IDPWLDAP* se denomina en el campo CONNAUTH del gestor de colas.

CSPUserIdDesplazamiento (MQLONG)
Es el desplazamiento en bytes del ID de usuario que se va a utilizar en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPUserIdLongitud (MQLONG)

Este campo es la longitud del ID de usuario que se utilizará en la autenticación.

La longitud máxima del ID de usuario depende de la plataforma, consulte ID de usuario. Si la longitud del ID de usuario es mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_NOT_AUTHORIZED.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

Reserved2 (MQBYTE8)

Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSPPasswordPtr (MQPTR)

Es la dirección en bytes de la contraseña que se va a utilizar en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

Este campo puede contener una contraseña vacía que es rechazada por el sistema operativo o la comprobación de contraseñas LDAP, en función de la configuración, pero no es rechazada por IBM MQ antes de pasarle el método de autenticación.

CSPPasswordOffset (MQLONG)

Es el desplazamiento en bytes de la contraseña que se va a utilizar en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPPasswordLength (MQLONG)

Este campo es la longitud de la contraseña que se utilizará en la autenticación.

La longitud máxima de la contraseña es MQ_CSP_PASSWORD_LENGTH, que es de 256 caracteres. Si la longitud de la contraseña es mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_NOT_AUTHORIZED. "

El valor de MQ_CSP_PASSWORD_LENGTH es 256.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

MQCTLO-Estructura de opciones de devolución de llamada de control

La estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control. La estructura es un parámetro de entrada y salida en la llamada MQCTL.

Disponibilidad

La estructura MQCTLO está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris

- Windows
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCTLO es MQCTLO_VERSION_1.

Juego de caracteres y codificación

Los datos de MQCTLO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 483. Campos en MQCTLO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucID (identificador de estructura)	MQCTLO_STRUC_ID	'CTLO'
Versión (número de versión de estructura)	MQCTLO_VERSION_1	1
Options (opciones)	MQCTLO_NONE	Nulos
Opciones (campo reservado)	Reservado, campo	
ConnectionArea (campo para que lo utilice la función de devolución de llamada)	Ninguna	Puntero nulo o bytes nulos

Notas:

1. En el lenguaje de programación C, la variable de macroMQCTLO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCTLO MyCTLO = {MQCTLO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
 };
```

Declaración COBOL para 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
```

Declaración PL/I para 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)

Estructura de opciones de control-Campo StrucId

Este es el identificador de estructura; el valor debe ser:

MQCTLO_STRUC_ID
: Identificador de la estructura de opciones de control.

  Para el lenguaje de programación C, también se define la constante MQCTLO_STRUC_ID_ARRAY; tiene el mismo valor que MQCTLO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCTLO_STRUC_ID.

Versión (MQLONG)

Estructura de opciones de control-Campo Versión

Este es el número de versión de la estructura; el valor debe ser:

MQCTLO_VERSION_1
: Version-1 Estructura de opciones de control.

La constante siguiente especifica el número de versión de la versión actual:

MQCTLO_CURRENT_VERSION
: Versión actual de la estructura de opciones de control.

Siempre es un campo de entrada. El valor inicial de este campo es MQCTLO_VERSION_1.

Opciones (MQLONG)

Estructura de opciones de control-Campo Opciones

Opciones que controlan la acción de MQCTL.

MQCTLO_FAIL_IF_QUIESCING
: Forzar que la llamada MQCTL falle si el gestor de colas o la conexión está en estado de desactivación temporal.

  Especifique MQGMO_FAIL_IF_QUIESCING, en las opciones MQGMO pasadas en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

MQCTLO_THREAD_AFFINITY
Esta opción informa al sistema de que la aplicación requiere que todos los consumidores de mensajes, para la misma conexión, se llamen en la misma hebra. Esta hebra se utilizará para todas las invocaciones de los consumidores hasta que se detenga la conexión.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQCTLO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQCTLO_NONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCTLO_NONE.

Reservado (MQLONG)

Este es un campo reservado. El valor debe ser cero.

ConnectionArea (MQPTR)

Estructura de opciones de control-Campo ConnectionArea

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios al campo ConnectionArea de la estructura MQCBC, que es un parámetro de entrada para la devolución de llamada.

Este campo se ignora para todas las operaciones que no sean MQOP_START y MQOP_START_WAIT.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

MQDH - Cabecera de distribución

La estructura MQDH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión. Un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales constan de la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR.Esta estructura la utilizan aplicaciones especializadas que colocan mensajes directamente en colas de transmisión o que eliminan mensajes de colas de transmisión (por ejemplo: agentes de canal de mensajes). Las aplicaciones que desean colocar mensajes en listas de distribución no deben utilizar esta estructura. En su lugar, deben utilizar la estructura MQOD para definir los destinos en la lista de distribución, y la estructura MQPMO para especificar propiedades de mensaje o recibir información sobre los mensajes enviados a los destinos individuales.

Disponibilidad

La estructura MQDH está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

MQFMT_DIST_HEADER

Juego de caracteres y codificación

Los datos de MQDH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de la MQDH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQDH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQDH (todos los demás casos).

Utilización

Cuando una aplicación coloca un mensaje en una lista de distribución, y algunos o todos los destinos son remotos, el gestor de colas prefija los datos del mensaje de aplicación con las estructuras MQXQH y MQDH, y coloca el mensaje en la cola de transmisión relevante. Por lo tanto, los datos se producen en la secuencia siguiente cuando el mensaje está en una cola de transmisión:

- estructura MQXQH
- Estructura MQDH más matrices de registros MQOR y MQPMR
- Datos de mensaje de aplicación

En función de los destinos, el gestor de colas puede generar más de un mensaje de este tipo y colocarlo en colas de transmisión diferentes. En este caso, las estructuras MQDH de esos mensajes identifican distintos subconjuntos de los destinos definidos por la lista de distribución abierta por la aplicación.

Una aplicación que coloca un mensaje de lista de distribución directamente en una cola de transmisión debe ajustarse a la secuencia descrita anteriormente y debe asegurarse de que la estructura MQDH sea correcta. Si la estructura MQDH no es válida, el gestor de colas puede fallar la llamada MQPUT o MQPUT1 con el código de razón MQRC_DH_ERROR.

Puede almacenar mensajes en una cola en formato de lista de distribución sólo si ha definido la cola como que puede dar soporte a los mensajes de lista de distribución. Consulte el atributo de cola **DistLists** descrito en “Atributos para colas” en la página 848. Si una aplicación coloca un mensaje de lista de distribución directamente en una cola que no da soporte a listas de distribución, el gestor de colas divide el mensaje de lista de distribución en mensajes individuales y, en su lugar, coloca los mensajes en la cola.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 484. Campos en MQDH para MQDH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQDH_STRUC_ID	'DH¬¬'
Versión (número de versión de estructura)	MQDH_VERSION_1	1
StrucLength (longitud de la estructura MQDH más los registros siguientes)	Ninguna	0
Codificación (codificación numérica de datos que sigue a la matriz de registros MQPMR)	Ninguna	0

Tabla 484. Campos en MQDH para MQDH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
CodedCharSetId (identificador de juego de caracteres de los datos que siguen a la matriz de registros MQPMR)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de los datos que siguen a la matriz de registros MQPMR)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos generales)	MQDHF_NONE	0
PutMsgRecFields (distintivos que indican qué campos MQPMR están presentes)	MQPMRF_NONE	0
RecsPresent (número de registros de objeto presentes)	Ninguna	0
ObjectRecDesplazamiento (desplazamiento del primer registro de objeto desde el inicio de MQDH)	Ninguna	0
PutMsgRecOffset (desplazamiento del primer registro de colocación de mensaje desde el inicio de MQDH)	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQDH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDH MyDH = {MQDH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de Visual Basic para 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)

El valor debe ser:

MQDH_STRUC_ID
Identificador de la estructura de cabecera de distribución.

Para el lenguaje de programación C, también se define la constante MQDH_STRUC_ID_ARRAY; tiene el mismo valor que MQDH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQDH_STRUC_ID.

Versión (MQLONG)

El valor debe ser:

MQDH_VERSION_1
Número de versión para la estructura de cabecera de distribución.

La constante siguiente especifica el número de versión de la versión actual:

MQDH_CURRENT_VERSION
Versión actual de la estructura de cabecera de distribución.

El valor inicial de este campo es MQDH_VERSION_1.

StrucLength (MQLONG)

Es el número de bytes desde el inicio de la estructura MQDH hasta el inicio de los datos de mensaje que siguen a las matrices de registros MQOR y MQPMR. Los datos se producen en la secuencia siguiente:

- estructura MQDH
- Matriz de registros MQOR
- Matriz de registros MQPMR
- Datos de mensaje

Las matrices de registros MQOR y MQPMR se direccionan mediante desplazamientos contenidos en la estructura MQDH. Si estos desplazamientos dan como resultado bytes no utilizados entre una o más de la estructura MQDH, las matrices de registros y los datos de mensaje, dichos bytes no utilizados deben incluirse en el valor de *StrucLength*, pero el gestor de colas no conserva el contenido de dichos bytes. Es válido que la matriz de registros MQPMR preceda a la matriz de registros MQOR.

El valor inicial de este campo es 0.

Encoding (MQLONG)

Es la codificación numérica de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos numéricos de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Es el identificador de juego de caracteres de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos de caracteres de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Es el nombre de formato de los datos que siguen a las matrices de registros MQOD y MQPMR (lo que ocurra en último lugar).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Puede especificar el distintivo siguiente:

MQDHF_NEW_MSG_IDS
Genere un nuevo identificador de mensaje para cada destino de la lista de distribución. Establézcalo sólo cuando no haya registros de colocación de mensajes o cuando los registros estén presentes pero no contengan el campo *MsgId* .

El uso de este distintivo difiere la generación de los identificadores de mensaje hasta el momento en que el mensaje de lista de distribución se divide finalmente en mensajes individuales. Esto minimiza la cantidad de información de control que debe fluir con el mensaje de lista de distribución.

Cuando una aplicación coloca un mensaje en una lista de distribución, el gestor de colas establece MQDHF_NEW_MSG_IDS en la MQDH que genera cuando se cumplen las dos sentencias siguientes:

- No hay registros de colocación de mensajes proporcionados por la aplicación, o los registros proporcionados no contienen el campo *MsgId* .
- El campo *MsgId* en MQMD es MQMI_NONE, o el campo *Options* en MQPMO incluye MQPMO_NEW_MSG_ID

Si no se necesitan distintivos, especifique lo siguiente:

MQDHF_NONE
No se han especificado distintivos. MQDHF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

El valor inicial de este campo es MQDHF_NONE.

PutMsgRecFields (MQLONG)

Puede especificar ninguno o varios de los distintivos siguientes:

MQPMRF_MSG_ID
El campo de identificador de mensaje está presente.

ID MQPMRF_CORREL_ID
El campo de identificador de correlación está presente.

MQPMRF_ID_grupo
El campo identificador de grupo está presente.

MQPMRF_FEEDBACK
El campo de comentarios está presente.

MQPMRF_ACCOUNTING_TOKEN
El campo de señal de contabilidad está presente.

Si no hay campos MQPMR presentes, especifique lo siguiente:

MQPMRF_NONE
No hay campos de registro de colocación de mensaje. MQPMRF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

El valor inicial de este campo es MQPMRF_NONE.

RecsPresent (MQLONG)

Es el número de destinos. Una lista de distribución siempre debe contener al menos un destino, por lo que *RecsPresent* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

Desplazamiento de ObjectRec(MQLONG)

Esto proporciona el desplazamiento en bytes del primer registro de la matriz de registros de objeto MQOR que contiene los nombres de las colas de destino. Hay *RecsPresent* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de objeto y el campo anterior) se incluyen en la longitud proporcionada por el campo *StrucLength* .

Una lista de distribución siempre debe contener al menos un destino, por lo que *ObjectRecOffset* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

PutMsgRecOffset (MQLONG)

Esto proporciona el desplazamiento en bytes del primer registro de la matriz de registros de mensajes de colocación MQPMR que contienen las propiedades del mensaje. Si está presente, hay *RecsPresent* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de mensaje de colocación y el campo anterior) se incluyen en la longitud proporcionada por el campo *StrucLength* .

Los registros de mensajes de colocación son opcionales; si no se proporcionan registros, *PutMsgRecOffset* es cero y *PutMsgRecFields* tiene el valor MQPMRF_NONE.

El valor inicial de este campo es 0.

MQDLH - Cabecera de mensajes no entregados

La estructura MQDLH describe la información que prefija los datos de mensaje de aplicación de los mensajes en la cola de mensajes no entregados (mensaje no entregado). Un mensaje puede llegar a la cola de mensajes no entregados ya sea porque el gestor de colas o el agente de canal de mensajes lo ha redirigido a la cola, o porque una aplicación ha colocado el mensaje directamente en la cola.

Nombre de formato

MQFMT_DEAD_LETTER_HEADER

Juego de caracteres y codificación

Los campos de la estructura MQDLH están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* . Estos se especifican en la estructura de cabecera que precede a la MQDLH, o en la estructura MQMD si la MQDLH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Si está utilizando las clases IBM MQ para Java/JMS, y la página de códigos definida en MQMD no está soportada por la máquina virtual Java , la MQDLH se escribe en el juego de caracteres UTF-8 .

Utilización

Las aplicaciones que colocan mensajes directamente en la cola de mensajes no entregados deben prefijar los datos del mensaje con una estructura MQDLH e inicializar los campos con los valores adecuados. Sin embargo, el gestor de colas no requiere que esté presente una estructura MQDLH, o que se hayan especificado valores válidos para los campos.

Si un mensaje es demasiado largo para colocarlo en la cola de mensajes no entregados, la aplicación debe realizar una de las acciones siguientes:

- Trunque los datos del mensaje para que quepan en la cola de mensajes no entregados.
- Anote el mensaje en el almacenamiento auxiliar y coloque un mensaje de informe de excepción en la cola de mensajes no entregados que lo indique.
- Descarte el mensaje y devuelva un error a su originador. Si el mensaje es (o puede ser) un mensaje crítico, hágalo sólo si se sabe que el originador todavía tiene una copia del mensaje; por ejemplo, un mensaje recibido por un agente de canal de mensajes de un canal de comunicación.

Cuál de las acciones anteriores es apropiada (si la hay) depende del diseño de la aplicación.

El gestor de colas realiza un proceso especial cuando un mensaje que es un segmento se coloca con una estructura MQDLH en la parte frontal; consulte la descripción de la estructura MQMDE para obtener más detalles.

Colocación de mensajes en la cola de mensajes no entregados

Cuando se coloca un mensaje en la cola de mensajes no entregados, la estructura MQMD utilizada para la llamada MQPUT o MQPUT1 debe ser idéntica al MQMD asociado con el mensaje (normalmente el MQMD devuelto por la llamada MQGET), con la excepción de lo siguiente:

- Establezca los campos *CodedCharSetId* y *Encoding* en cualquier juego de caracteres y codificación que se utilicen para los campos de la estructura MQDLH.
- Establezca el campo *Format* en MQFMT_DEAD_LETTER_HEADER para indicar que los datos empiezan por una estructura MQDLH.
- Establezca los campos de contexto (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime*, *UserIdentifier*) utilizando una opción de contexto adecuada a las circunstancias:
  - Una aplicación que coloca en la cola de mensajes no entregados un mensaje que no está relacionado con ningún mensaje anterior debe utilizar la opción MQPMO_DEFAULT_CONTEXT; esto hace que el gestor de colas establezca todos los campos de contexto del descriptor de mensaje en sus valores predeterminados.
  - Una aplicación de servidor que pone en la cola de mensajes no entregados un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_ALL_CONTEXT para conservar la información de contexto original.
  - Una aplicación de servidor que coloca en la cola de mensajes no entregados una *respuesta* a un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_IDENTITY_CONTEXT; esto

conserva la información de identidad pero establece la información de origen en la de la aplicación de servidor.

– Un agente de canal de mensajes que coloca en la cola de mensajes no entregados un mensaje que ha recibido de su canal de comunicación debe utilizar la opción MQPMO_SET_ALL_CONTEXT para conservar la información de contexto original.

En la propia estructura MQDLH, establezca los campos como se indica a continuación:

- Establezca los campos *CodedCharSetId*, *Encoding*y *Format* en los valores que describen los datos que siguen a la estructura MQDLH, normalmente los valores del descriptor de mensaje original.
- Establezca los campos de contexto *PutApplType*, *PutApplName*, *PutDate*y *PutTime* en valores adecuados para la aplicación que coloca el mensaje en la cola de mensajes no entregados; estos valores no están relacionados con el mensaje original.
- Establezca otros campos según corresponda.

Asegúrese de que todos los campos tienen valores válidos y que los campos de caracteres se rellenan con espacios en blanco hasta la longitud definida del campo; no finalice los datos de caracteres de forma prematura utilizando un carácter nulo, porque el gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQDLH.

Obtención de mensajes de la cola de mensajes no entregados

Las aplicaciones que obtienen mensajes de la cola de mensajes no entregados deben verificar que los mensajes empiezan con una estructura MQDLH. La aplicación puede determinar si una estructura MQDLH está presente examinando el campo *Format* en el descriptor de mensaje MQMD; si el campo tiene el valor MQFMT_DEAD_LETTER_HEADER, los datos del mensaje empiezan por una estructura MQDLH. Tenga en cuenta también que los mensajes que las aplicaciones obtienen de la cola de mensajes no entregados pueden truncarse si originalmente eran demasiado largos para la cola.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 485. Campos en MQDLH para MQDLH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQDLH_STRUC_ID	'DLH¬'
Versión (número de versión de estructura)	MQDLH_VERSION_1	1
Razón (el mensaje de razón ha llegado a la cola de mensajes no entregados)	MQRC_NONE	0
DestQName (nombre de la cola de destino original)	Ninguna	Serie nula o espacios en blanco
DestQMgrNombre (nombre del gestor de colas de destino original)	Ninguna	Serie nula o espacios en blanco
Codificación (codificación numérica de datos que sigue a MQDLH)	Ninguna	0
CodedCharSetId (identificador de juego de caracteres de los datos que siguen a MQDLH)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de los datos que siguen a MQDLH)	MQFMT_NONE	Espacios en blanco

Tabla 485. Campos en MQDLH para MQDLH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
TipoPutAppl (tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados)	Ninguna	0
PutApplNombre (nombre de la aplicación que coloca el mensaje en la cola de mensajes no entregados)	Ninguna	Serie nula o espacios en blanco
PutDate (fecha en la que se colocó el mensaje en la cola de mensajes no entregados)	Ninguna	Serie nula o espacios en blanco
PutTime (hora a la que se colocó el mensaje en la cola de mensajes no entregados)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQDLH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

StrucId es el identificador de estructura.

El valor debe ser:

MQDLH_STRUC_ID
Identificador de la estructura de cabecera de mensaje no entregado.

Para el lenguaje de programación C, también se define la constante MQDLH_STRUC_ID_ARRAY; tiene el mismo valor que MQDLH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQDLH_STRUC_ID.

Versión (MQLONG)

La versión es el número de versión de la estructura.

El valor debe ser:

MQDLH_VERSION_1
Número de versión para la estructura de cabecera de mensaje no entregado.

La constante siguiente especifica el número de versión de la versión actual:

MQDLH_CURRENT_VERSION
Versión actual de la estructura de cabecera de mensaje no entregado.

El valor inicial de este campo es MQDLH_VERSION_1.

Razón (MQLONG)

El campo Razón identifica la razón por la que el mensaje se ha colocado en la cola de mensajes no entregados en lugar de en la cola de destino original.

Esto identifica la razón por la que el mensaje se ha colocado en la cola de mensajes no entregados en lugar de en la cola de destino original. Debe ser uno de los valores MQFB_ * o MQRC_ * (por ejemplo,

MQRC_Q_FULL). Consulte la descripción del campo *Feedback* en “MQMD - Descriptor de mensaje” en la página 421 para obtener detalles de los valores MQFB_ * comunes que se pueden producir.

Si el valor está en el rango de MQFB_IMS_FIRST a MQFB_IMS_LAST, el código de error IMS real se puede determinar restando MQFB_IMS_ERROR del valor del campo *Reason* .

Algunos valores MQFB_ * sólo aparecen en este campo. Están relacionados con mensajes de repositorio, mensajes desencadenantes o mensajes de cola de transmisión que se han transferido a la cola de mensajes no entregados. Son las siguientes:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')
Una aplicación que procesa un mensaje desencadenante no puede iniciar la aplicación especificada en el campo *ApplId* del mensaje desencadenante (consulte “MQTM-Mensaje de desencadenante” en la página 606).

En z/OS, la transacción CICS de CKTI es un ejemplo de una aplicación que procesa mensajes desencadenantes.

MQFB_APPL_TYPE_ERROR (X'0000010B')
Una aplicación que procesa un mensaje desencadenante no puede iniciar la aplicación porque el campo *ApplType* del mensaje desencadenante no es válido (consulte “MQTM-Mensaje de desencadenante” en la página 606).

En z/OS, la transacción CICS de CKTI es un ejemplo de una aplicación que procesa mensajes desencadenantes.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')
El mensaje estaba en SYSTEM.CLUSTER.TRANSMIT.QUEUE está pensado para una cola de clúster que se ha abierto con la opción MQOO_BIND_ON_OPEN, pero el canal de clúster receptor remoto que se debe utilizar para transmitir el mensaje a la cola de destino se ha suprimido antes de que se pudiera enviar el mensaje. Debido a que se ha especificado MQOO_BIND_ON_OPEN, sólo puede utilizarse el canal seleccionado cuando se abrió la cola para transmitir el mensaje. Como este canal ya no está disponible, el mensaje se coloca en la cola de mensajes no entregados.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')
El mensaje no es un mensaje de repositorio.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')
La salida de definición automática de canal ha detenido el mensaje.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')
El mensaje ha sido detenido por la salida de mensajes de canal.

MQFB_TM_ERROR (X'0000010A')
El campo *Format* en MQMD especifica MQFMT_TRIGGER, pero el mensaje no empieza por una estructura MQTM válida. Por ejemplo, es posible que el captador de atención mnemotécnico de *StrucId* no sea válido, que no se reconozca *Version* o que la longitud del mensaje desencadenante sea insuficiente para contener la estructura MQTM.

En z/OS, la transacción CICS de CKTI es un ejemplo de una aplicación que procesa mensajes desencadenantes y puede generar este código de comentarios.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')
Un agente de canal de mensajes ha detectado que un mensaje de la cola de transmisión no tiene el formato correcto. El agente de canal de mensajes coloca el mensaje en la cola de mensajes no entregados utilizando este código de retorno.

Una causa común es que un mensaje se ha colocado directamente en la cola de transmisión, por lo que el mensaje no tiene la cabecera XQH esperada. Los mensajes deben colocarse en una cola de transmisión a través de una cola remota, a menos que la aplicación cree la cabecera MQXQH.

El valor inicial de este campo es MQRC_NONE.

DestQName (MQCHAR48)

DestQName es el nombre de la cola de mensajes que era el destino original del mensaje.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

DestQMgrNombre (MQCHAR48)

DestQMgrNombre es el nombre del gestor de colas que era el destino original del mensaje.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Encoding (MQLONG)

La codificación es la codificación numérica de los datos que siguen a la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos numéricos de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es `0`.

CodedCharSetId (MQLONG)

CodedCharSetId es el identificador de juego de caracteres de los datos que fluyen a través de la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos de tipo carácter de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT

Los datos de caracteres de los datos que siguen a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

El formato es el nombre de formato de los datos que siguen a la estructura MQDLH (normalmente los datos del mensaje original).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

PutApplType (MQLONG)

PutApplTipo es el tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados.

Este campo tiene el mismo significado que el campo *PutApplType* en el descriptor de mensaje MQMD (consulte "MQMD - Descriptor de mensaje" en la página 421 para obtener detalles).

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplType* tiene el valor MQAT_QMGR.

El valor inicial de este campo es 0.

PutApplNombre (MQCHAR28)

PutApplNombre es el nombre de la aplicación que ha colocado el mensaje en la cola de mensajes no entregados (undelivered-message).

El formato del nombre depende del campo *PutApplType* . El formato puede variar de un release a otro. Consulte la descripción del campo *PutApplName* en "MQMD - Descriptor de mensaje" en la página 421.

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplName* contiene los primeros 28 caracteres del nombre del gestor de colas, rellenados con espacios en blanco si es necesario.

La longitud de este campo la proporciona MQ_PUT_APPL_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 28 caracteres en blanco en otros lenguajes de programación.

PutDate (MQCHAR8)

PutDate es la fecha en la que el mensaje se colocó en la cola de mensajes no entregados.

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- AAAAMMDD

donde los caracteres representan:

AAAA
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona MQ_PUT_DATE_LENGTH. El valor inicial de este campo es la serie nula en C y ocho caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)

PutTime es la hora a la que se ha colocado el mensaje en la cola de mensajes no entregados.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan:

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte la nota)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, es posible que en raras ocasiones se devuelvan 60 o 61 durante los segundos en *PutTime*. Esto sucede cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la serie nula en C y ocho caracteres en blanco en otros lenguajes de programación.

MQDMHO-Suprimir opciones de manejador de mensajes

La estructura **MQDMHO** permite a las aplicaciones especificar opciones que controlan cómo se suprimen los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada **MQDLTMH** .

Juego de caracteres y codificación

Los datos de **MQDMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 486. Campos en MQDMHO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQDMHO_STRUC_ID	'DMHO'
Versión (número de versión de estructura)	MQDMHO_VERSION_1	1
Options (opciones)	MQDMHO_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQDMHO_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQDMHO

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
```

```
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQDLTMH */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este es el identificador de estructura; el valor debe ser:

MQDMHO_STRUC_ID

Identificador de la estructura de opciones de manejador de mensajes de supresión.

Para el lenguaje de programación C, la constante **MQDMHO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQDMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_STRUC_ID**.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQDMHO_VERSION_1
Version-1 suprime la estructura de opciones del manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

MQDMHO_CURRENT_VERSION
Versión actual de la estructura de opciones de manejador de mensajes de supresión.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_VERSION_1**.

Opciones (MQLONG)

El valor debe ser:

MQDMHO_NONE

No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_NONE**.

MQDMPO-Suprimir opciones de propiedad de mensaje

La estructura MQDMPO permite a las aplicaciones especificar opciones que controlan cómo se suprimen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Juego de caracteres y codificación

Los datos de MQDMPO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 487. Campos en MQDPMO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQDMPO_STRUC_ID	'DMPO'
Versión (número de versión de estructura)	MQDMPO_VERSION_1	1
Options (opciones que controlan la acción de MQDMPO)	Opciones que controlan la acción de MQDLTMP	MQDMPO_NONE

Notas:

1. En el lenguaje de programación C, la variable de macroMQDMPO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Suprimir estructura de opciones de propiedad de mensaje-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQDMPO_STRUC_ID
Identificador para la estructura de opciones de propiedad de mensaje de supresión.

Para el lenguaje de programación C, también se define la constante MQDMPO_STRUC_ID_ARRAY; tiene el mismo valor que MQDMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_STRUC_ID.

Versión (MQLONG)

Suprimir estructura de opciones de propiedad de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQDMPO_VERSION_1
Número de versión para la estructura de opciones de suprimir propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQDMPO_CURRENT_VERSION
Versión actual de la estructura de opciones de supresión de propiedades de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_VERSION_1.

Opciones (MQLONG)

Suprimir estructura de opciones de propiedad de mensaje-Campo Opciones

Opciones de ubicación: Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de la propiedad.

MQDMPO_DEL_FIRST
Suprime la primera propiedad que coincide con el nombre especificado.

MQDMPO_DEL_PROP_UNDER_CURSOR
Suprime la propiedad a la que apunta el cursor de propiedad; es decir, la propiedad que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o la opción MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje. También se restablece cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO en una llamada MQGET, o estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE. Si la propiedad a la que apunta el cursor de propiedad ya se ha suprimido, la llamada también falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE.

Si no se requiere ninguna de las opciones, se puede utilizar la siguiente opción:

MQDMPO_NONE
No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_DEL_FIRST.

MQEPH - Cabecera PCF incrustada

La estructura MQEPH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de formato de mandato programable (PCF). El campo *PCFHeader* define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otras cabeceras.

Nombre de formato

MQFMT_EMBEDDED_PCF

Juego de caracteres y codificación

Los datos de MQEPH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de MQEPH en los campos *CodedCharSetId* y *Encoding* en MQMD (si la estructura MQEPH está al principio de los datos del mensaje), o la estructura de cabecera que precede a la estructura MQEPH (todos los demás casos).

Utilización

No puede utilizar estructuras MQEPH para enviar mandatos al servidor de mandatos o a cualquier otro servidor que acepte PCF del gestor de colas.

De forma similar, el servidor de mandatos o cualquier otro servidor de aceptación de PCF de gestor de colas no generan respuestas o sucesos que contengan estructuras MQEPH.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 488. Campos en MQEPH para MQEPH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQEPH_STRUC_ID	'EPH¬'
Versión (número de versión de estructura)	MQEPH_VERSION_1	1
StrucLength (longitud de la estructura MQEPH más MQCFH y estructuras de parámetros que le siguen)	MQEPH_STRUC_LENGTH_FIXED	68
Codificación (codificación numérica de datos que sigue a la última estructura de parámetros PCF)	Ninguna	0
CodedCharSetId (identificador de juego de caracteres de los datos que siguen a la última estructura de parámetro PCF)	MQCCSI_UNDEFINED	0

Tabla 488. Campos en MQEPH para MQEPH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
Formato (nombre de formato de los datos que siguen a la estructura del último parámetro PCF)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQEPH_NONE	0
PCFHeader (cabecera PCF (formato de mandato programable))	Nombres y valores tal como se definen en Tabla 489 en la página 366	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQEPH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
 };
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

El valor debe ser:

MQEPH_STRUC_ID
Identificador de la estructura de cabecera de distribución.

Para el lenguaje de programación C, también se define la constante MQEPH_STRUC_ID_ARRAY; tiene el mismo valor que MQDH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQEPH_STRUC_ID.

Versión (MQLONG)

El valor debe ser:

MQEPH_VERSION_1
Número de versión para la estructura de cabecera PCF incorporada.

La constante siguiente especifica el número de versión de la versión actual:

MQCFH_VERSION_3
Versión actual de la estructura de cabecera PCF incorporada.

El valor inicial de este campo es MQEPH_VERSION_1.

StrucLength (MQLONG)

Es la cantidad de datos que preceden a la siguiente estructura de cabecera. Esto incluye:

- Longitud de la cabecera MQEPH
- La longitud de todos los parámetros PCF que siguen a la cabecera
- Cualquier relleno en blanco que siga a estos parámetros

StrucLength debe ser un múltiplo de 4.

La parte de longitud fija de la estructura está definida por MQEPH_STRUC_LENGTH_FIXED.

El valor inicial de este campo es 68.

Encoding (MQLONG)

Es la codificación numérica de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Es el identificador de juego de caracteres de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Es el nombre de formato de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Están disponibles los valores siguientes:

MQEPH_NONE
No se han especificado distintivos. MQEPH_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

MQEPH_CCSID_EMBEDDED
El juego de caracteres de los parámetros que contienen datos de caracteres se especifica individualmente en el campo CodedCharSetId de cada estructura. El juego de caracteres de los campos StrucId y Format se define mediante el campo CodedCharSetId en la estructura de cabecera que precede a la estructura MQEPH, o mediante el campo CodedCharSetId en MQMD si MQEPH está al principio del mensaje.

El valor inicial de este campo es MQEPH_NONE.

PCFHeader (MQCFH)

Es la cabecera de formato de mandato programable (PCF), que define los parámetros PCF que siguen a la estructura MQEPH. Esto le permite seguir los datos del mensaje PCF con otras cabeceras.

La cabecera PCF se define inicialmente con los valores siguientes:

Tabla 489. Valores iniciales de los campos en MQCFH

Nombre de campo	Nombre de constante	Valor de constante
Type	MQCFT_NONE	0
StrucLength	MQCFH_STRUC_LENGTH	36
Version	MQCFH_VERSION_3	3
StrucLength	Ninguna	0
Command	MQCMD_NONE	0
MsgSeqNumber	Ninguna	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	Ninguna	0

La aplicación debe cambiar Type de MQCFT_NONE a un tipo de estructura válido para el uso que está haciendo de la cabecera PCF incorporada.

MQGMO-Opciones de obtención de mensajes

La estructura MQGMO permite a la aplicación controlar cómo se eliminan los mensajes de las colas. La estructura es un parámetro de entrada/salida en la llamada MQGET.

Versión

La versión actual de MQGMO es MQGMO_VERSION_4. Determinados campos sólo están disponibles en determinadas versiones de MQGMO. Si necesita portar aplicaciones entre varios entornos, debe asegurarse de que la versión de MQGMO sea coherente en todos los entornos. Los campos que existen sólo en determinadas versiones de la estructura se identifican como tales en "MQGMO-Opciones de obtención de mensajes" en la página 367 y en las descripciones de campo.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQGMO soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQGMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , establezca el campo *Version* en el número de versión de la versión necesaria.

Juego de caracteres y codificación

Los datos de MQGMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 490. Campos en MQGMO para MQGMO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQGMO_STRUC_ID	'GMO¬'
Versión (número de versión de estructura)	MQGMO_VERSION_1	1
MQGMO-Campo Options (opciones que controlan la acción de MQGET)	MQGMO_NO_WAIT	0
WaitInterval (intervalo de espera)	Ninguna	0
Signal1 (señal)	Ninguna	Puntero nulo en z/OS ; 0 de lo contrario
Signal2 (identificador de señal)	Ninguna	0
ResolvedQName (nombre resuelto de la cola de destino)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_2.		
MatchOptions (opciones que controlan los criterios de selección utilizados para MQGET)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (distintivo que indica si el mensaje recuperado está en un grupo)	MQGS_NOT_IN_GROUP	'¬'

Tabla 490. Campos en MQGMO para MQGMO (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
SegmentStatus (distintivo que indica si el mensaje recuperado es un segmento de un mensaje lógico)	MQSS_NO_A_SEGMENTO	'¬'
Segmentación (distintivo que indica si se permite una segmentación adicional para el mensaje recuperado)	MQSEG_INHIBIDO	'¬'
Reserved1 (reservado)	Ninguna	'¬'
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_3.		
MsgToken (señal de mensaje)	MQMTOK_NONE	Nulos
ReturnedLength (longitud en bytes de datos de mensaje devueltos)	MQRL_UNDEFINED	-1
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_4.		
Reserved2 (reservado)	Ninguna	'¬'
MsgHandle (descriptor de contexto de un mensaje que se va a rellenar con las propiedades del mensaje que se está recuperando de la cola)	MQHM_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQGMO_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQGMO

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of */
                                MQGET */
  MQLONG    WaitInterval;    /* Wait interval */
  MQLONG    Signal1;         /* Signal */
  MQLONG    Signal2;         /* Signal identifier */
  MQCHAR48  ResolvedQName;   /* Resolved name of destination queue */
  /* Ver:1 */
  MQLONG    MatchOptions;    /* Options controlling selection */
                                criteria used for MQGET */
  MQCHAR    GroupStatus;     /* Flag indicating whether message */
                                retrieved is in a group */
  MQCHAR    SegmentStatus;   /* Flag indicating whether message */
                                retrieved is a segment of a logical */
                                message */
  MQCHAR    Segmentation;    /* Flag indicating whether further */
                                segmentation is allowed for the */
                                message retrieved */
```

```
  MQCHAR    Reserved1;       /* Reserved */
  /* Ver:2 */
  MQBYTE16  MsgToken;        /* Message token */
  MQLONG    ReturnedLength;  /* Length of message data returned */
                                (bytes) */
  /* Ver:3 */
  MQLONG    Reserved2;       /* Reserved */
  MQHMSG    MsgHandle;       /* Message handle */
  /* Ver:4 */
};
```

Nota: En z/OS, el campo *Signal1* se declara como PMQLONG.

Declaración COBOL para MQGMO

```
**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4).
**    Structure version number
   15 MQGMO-VERSION        PIC S9(9) BINARY.
**    Options that control the action of MQGET
   15 MQGMO-OPTIONS        PIC S9(9) BINARY.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) BINARY.
**    Signal
   15 MQGMO-SIGNAL1        PIC S9(9) BINARY.
**    Signal identifier
   15 MQGMO-SIGNAL2        PIC S9(9) BINARY.
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48).
**    Options controlling selection criteria used for MQGET
   15 MQGMO-MATCHOPTIONS   PIC S9(9) BINARY.
**    Flag indicating whether message retrieved is in a group
   15 MQGMO-GROUPSTATUS    PIC X.
**    Flag indicating whether message retrieved is a segment of a
**    logical message
   15 MQGMO-SEGMENTSTATUS  PIC X.
**    Flag indicating whether further segmentation is allowed for the
**    message retrieved
   15 MQGMO-SEGMENTATION   PIC X.
**    Reserved
   15 MQGMO-RESERVED1      PIC X.
**    Message token
   15 MQGMO-MSGTOKEN       PIC X(16).
**    Length of message data returned (bytes)
   15 MQGMO-RETURNEDLENGTH PIC S9(9) BINARY.
**    Reserved
   15 MQGMO-RESERVED2      PIC S9(9) BINARY.
**    Message handle
   15 MQGMO-MSGHANDLE      PIC S9(18) BINARY.
```

Nota: En z/OS, el campo *Signal1* se declara como POINTER.

Declaración PL/I para MQGMO

```
dcl
 1 MQGMO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of
                                     MQGET */
  3 WaitInterval   fixed bin(31), /* Wait interval */
  3 Signal1        fixed bin(31), /* Signal */
  3 Signal2        fixed bin(31), /* Signal identifier */
  3 ResolvedQName  char(48),      /* Resolved name of destination
                                     queue */
  3 MatchOptions   fixed bin(31), /* Options controlling selection
                                     criteria used for MQGET */
  3 GroupStatus    char(1),       /* Flag indicating whether message
                                     retrieved is in a group */
  3 SegmentStatus  char(1),       /* Flag indicating whether message
                                     retrieved is a segment of a logical
                                     message */
  3 Segmentation   char(1),       /* Flag indicating whether further
                                     segmentation is allowed for the
                                     message retrieved */
```

```
  3 Reserved1      char(1),       /* Reserved */
  3 MsgToken       char(16),      /* Message token */
  3 ReturnedLength fixed bin(31); /* Length of message data returned
                                     (bytes) */
  3 Reserved2      fixed bin(31); /* Reserved */
  3 MsgHandle      fixed bin(63); /* Message handle */
```

Nota: En z/OS, el campo *Signal1* se declara como `pointer`.

Declaración de High Level Assembler para 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)
```

Declaración de High Level Assembler para 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
```

Opciones de canal PROPCTL para MQGMO

Utilice el atributo de canal **PROPCTL** para controlar qué propiedades de mensaje se incluyen en un mensaje que se envía desde un gestor de colas IBM MQ 9.1 a un gestor de colas asociado desde una versión anterior de IBM MQ.

Tabla 491. Valores de atributos de propiedades de mensaje de canal	
PROPCTL	**Descripción**
todos	Utilice esta opción si las aplicaciones conectadas al gestor de colas asociado de una versión anterior pueden procesar las propiedades colocadas en un mensaje por una aplicación IBM MQ 9.1 . Todas las propiedades se envían al gestor de colas asociado, además de los pares nombre-valor colocados en MQRFH2. Debe contemplar dos posibilidades de diseño de aplicaciones: 1. Una aplicación conectada al gestor de colas asociado debe poder procesar mensajes que contengan cabeceras MQRFH2 generadas en un gestor de colas IBM MQ 9.1 . 2. La aplicación conectada al gestor de colas de asociado debe procesar nuevas propiedades de mensaje que están marcadas con MQPD_SUPPORT_REQUIRED correctamente. Con la opción de canal ALL establecida, las aplicaciones JMS pueden interoperar entre IBM MQ 9.1 y una versión anterior utilizando el canal. Las nuevas aplicaciones IBM MQ 9.1 que utilizan propiedades de mensaje pueden interactuar con aplicaciones de una versión anterior, dependiendo de cómo la aplicación de la versión anterior gestione las cabeceras MQRFH2.

COMPAT	Utilice esta opción para enviar propiedades de mensaje a aplicaciones conectadas a un gestor de colas de asociado de una versión anterior en algunos casos, pero no siempre. Las propiedades de mensaje sólo se envían si se cumplen dos condiciones:
	
	1. Ninguna propiedad debe marcarse de modo que requiera proceso de propiedad de mensaje.
	2. Una de las propiedades de mensaje debe estar como mínimo en una carpeta "reservada"; consulte Nota.
	
	Con la opción de canal COMPAT establecida, las aplicaciones JMS pueden interoperar entre IBM MQ 9.1 y una versión anterior utilizando el canal.
	
	El canal no está disponible para cada aplicación utilizando propiedades de mensaje, sólo para aquellas aplicaciones que utilizan las carpetas reservadas. Las reglas relativas a si el mensaje o la propiedad se envía son:
	
	1. Si el mensaje tiene propiedades, pero ninguna de las propiedades está asociada con una carpeta "reservada", no se envía ninguna propiedad de mensaje.
	2. Si se ha creado cualquier propiedad de mensaje en una carpeta de propiedad "reservada", se envían todas las propiedades de mensaje asociadas con el mensaje. No obstante:
	  a. Si alguna de las propiedades de mensaje está marcada para requerir soporte, MQPD_SUPPORT_REQUIRED o MQPD_SUPPORT_REQUIRED_IF_LOCAL, se rechaza todo el mensaje. Se devuelve, descarga o envía a la cola de mensajes no entregados según el valor de sus opciones de informe.
	  b. Si no hay ninguna propiedad de mensaje que requiera soporte, es posible que una propiedad individual no se envíe. Si alguno de los campos de descriptor de propiedad de mensaje está establecido en valores no predeterminados, la propiedad individual no se envía. El mensaje se envía de todos modos. Un ejemplo de valor de campo de descriptor de propiedad no predeterminado es MQPD_USER_CONTEXT.
	
	Nota: Los nombres de carpetas "reservadas" empiezan por mcd., jms., usr. o mqext.. Estas carpetas se crean para aplicaciones que utilizan la interfaz JMS. En IBM MQ 9.1 , los pares de nombre-valor que se colocan en estas carpetas se tratan como propiedades de mensaje.
	
	Las propiedades de mensaje se envían en una cabecera MQRFH2 , además de los pares nombre-valor colocados en una cabecera MQRFH2 . Los pares de nombre-valor colocados en una cabecera MQRFH2 se envían siempre que el mensaje no se rechace.

Tabla 491. Valores de atributos de propiedades de mensaje de canal (continuación)	
PROPCTL	**Descripción**
NONE	Utilice esta opción para impedir que se envíe cualquier propiedad de mensaje a aplicaciones conectadas a un gestor de colas de asociado de una versión anterior. Un MQRFH2 que contiene pares de nombre-valor y propiedades de mensaje se sigue enviando, pero sólo con los pares de nombre-valor. Con la opción de canal NONE establecida, se envía un mensaje JMS como JMSTextMessage o un JMSBytesMessage sin ninguna propiedad de mensaje JMS. Si es posible que una aplicación de una versión anterior ignore todas las propiedades establecidas en una aplicación IBM MQ 9.1 , puede interactuar con ella.

Opciones de cola de PROPCTL para MQGMO

Utilice el atributo de cola **PROPCTL** para controlar cómo se devuelven las propiedades de mensaje a una aplicación que llama a **MQGET** sin establecer ninguna opción de propiedad de mensaje **MQGMO**.

Tabla 492. Valores de atributos de propiedades de mensaje de cola	
PROPCTL	**Descripción**
todos	Utilice la opción ALL para que distintas aplicaciones que lean un mensaje de la misma cola puedan procesar el mensaje de distintas maneras. • Una aplicación migrada sin cambios desde una versión anterior puede leer la cabecera MQRFH2 directamente. Las propiedades son accesibles directamente en la cabecera MQRFH2. Debe modificar la aplicación para manejar cualquier propiedad nueva y atributos de propiedad nuevos. Es posible que la aplicación se pueda ver afectada por cambios en el diseño y el número de cabeceras MQRFH2. Algunos atributos de carpeta pueden ser eliminados o puede que IBM MQ notifique un error en el diseño de la cabecera MQRFH2 que pasó por alto en una versión anterior. • Una aplicación nueva o modificada puede utilizar la MQI de propiedad de mensaje para consultar propiedades de mensaje y leer pares de nombre-valor en la cabecera MQRFH2 directamente. Todas las propiedades del mensaje se devuelven a la aplicación. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Los pares de nombre-valor que no son propiedades de mensaje permanecen en MQRFH2, que se elimina de cualquier propiedad de mensaje. • Si la aplicación no crea un manejador de mensajes, todas las propiedades de mensaje y pares de nombre-valor permanecen en MQRFH2. ALL sólo tiene este efecto si la aplicación receptora no ha establecido una opción MQGMO_PROPERTIES o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.

Tabla 492. Valores de atributos de propiedades de mensaje de cola (continuación)	
PROPCTL	**Descripción**
COMPAT (valor predeterminado)	COMPAT es la opción predeterminada. Si GMO_PROPERTIES_* no se ha establecido, como en una aplicación no modificada de una versión anterior, se presupone COMPAT. Mediante el uso de COMPAT como opción predeterminada, una aplicación de una versión anterior que no creó explícitamente una cabecera MQRFH2, puede funcionar sin cambios en IBM MQ 9.1. Utilice esta opción si ha escrito una aplicación MQI de una versión anterior para leer mensajes JMS. • Las propiedades JMS , que se almacenan en una cabecera MQRFH2 , se devuelven a la aplicación en una cabecera MQRFH2 en carpetas con nombres que empiezan por mcd., jms., usr. o mqext. • Si el mensaje tiene carpetas JMS y si una aplicación IBM MQ 9.1 añade nuevas carpetas de propiedades al mensaje, estas propiedades también se devuelven en MQRFH2. Por lo tanto, debe modificar la aplicación para manejar cualquier propiedad nueva y nuevos atributos de propiedad. Es posible que una aplicación no modificada se pueda ver afectada por cambios en el diseño y el número de cabeceras MQRFH2. Puede encontrar que algunos atributos de carpeta se han eliminado o que IBM MQ encuentra errores en el diseño de la cabecera MQRFH2 que se pasan por alto en una versión anterior. **Nota:** En este caso, el comportamiento de la aplicación es el mismo tanto si está conectada a un gestor de colas de una versión anterior o IBM MQ 9.1. Si el atributo de canal **PROPCTL** está establecido en COMPAT o ALL, todas las propiedades de mensaje nuevas se envían al gestor de colas de la versión anterior. • Si el mensaje no es un mensaje JMS, pero contiene otras propiedades, dichas propiedades no se devuelven a la aplicación en una cabecera MQRFH2.[1] • Esta opción también permite que, en muchos casos, las aplicaciones de una versión anterior que crean explícitamente una cabecera MQRFH2 funcionen correctamente. Por ejemplo, un programa de la interfaz de colas de mensajes que crea una MQRFH2 que contiene propiedades de mensaje JMS continúa funcionando correctamente. Si un mensaje se crea sin propiedades de mensaje JMS pero con algunas otras carpetas MQRFH2, las carpetas se devuelven a la aplicación. Sólo se eliminarán dichas carpetas específicas de la MQRFH2 si son carpetas de propiedades de mensaje. Las carpetas de propiedades de mensaje se identifican por tener el nuevo atributo de carpeta content='properties', o son carpetas cuyo nombre aparece en Nombre de carpeta de propiedades definidas o Nombre de carpeta de propiedades sin agrupar. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Las propiedades de mensaje se eliminan de las cabeceras MQRFH2. Los pares de nombre-valor que no son propiedades de mensaje permanecen en MQRFH2. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, puede consultar a todas las propiedades de mensaje, independientemente de si el mensaje tiene carpetas JMS. • Si la aplicación no crea un manejador de mensajes, todas las propiedades de mensaje y pares de nombre-valor permanecen en MQRFH2. Si un mensaje contiene nuevas carpetas de propiedades de usuario, puede deducir que el mensaje lo creó una aplicación IBM MQ 9.1 nueva o cambiada. Si la aplicación receptora va a procesar estas propiedades directamente en una MQRFH2, debe modificar la aplicación para utilizar la opción ALL. Con la opción predeterminada COMPAT establecida, una aplicación no modificada puede procesar el resto de la cabecera MQRFH2, sin las propiedades de IBM MQ 9.1. El objetivo de la interfaz PROPCTL es dar soporte a las aplicaciones antiguas que leen carpetas MQRFH2 y a las aplicaciones nuevas y cambiadas utilizando la interfaz de propiedades de mensaje. Está pensada para que las nuevas aplicaciones utilicen la interfaz de propiedades de mensaje para todas las propiedades de mensaje de usuario y para evitar leer y escribir cabeceras MQRFH2 directamente. COMPAT [illegible]

Tabla 492. Valores de atributos de propiedades de mensaje de cola (continuación)	
PROPCTL	**Descripción**
FORCE	La opción FORCE coloca todas las propiedades de mensaje en cabeceras MQRFH2. Todas las propiedades de mensaje y los pares de nombre-valor de las cabeceras MQRFH2 permanecen en el mensaje. Las propiedades de mensaje no se eliminan de MQRFH2 y están disponibles a través de un manejador de mensajes. El efecto de elegir la opción FORCE es habilitar una aplicación recién migrada para leer propiedades de mensaje de cabeceras MQRFH2. Suponga que ha modificado una aplicación para procesar propiedades de mensaje de la IBM MQ 9.1 pero también ha mantenido la posibilidad de que funcione directamente con cabeceras MQRFH2, como se hacía anteriormente. Puede decidir cuándo la aplicación pasará a utilizar propiedades de mensaje estableciendo inicialmente el atributo de cola PROPCTL en FORCE. Establezca el atributo de cola **PROPCTL** en otro valor cuando esté preparado para empezar a utilizar propiedades de mensaje. Si la nueva función en la aplicación no se comporta según lo previsto, establezca la opción **PROPCTL** en FORCE. FORCE sólo tiene este efecto si la aplicación receptora no ha establecido una opción MQGMO_PROPERTIES o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.
NONE	Utilice la opción NONE para que una aplicación existente pueda procesar un mensaje, ignorando todas las propiedades del mensaje y una aplicación nueva o modificada pueda consultar las propiedades del mensaje. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Los pares de nombre-valor que no son propiedades de mensaje permanecen en MQRFH2, que se elimina de cualquier propiedad de mensaje. • Si la aplicación no crea un manejador de mensajes, todas las propiedades de mensaje se eliminan de MQRFH2. Los pares de nombre-valor en las cabeceras MQRFH2 permanecen en el mensaje. NONE sólo tiene este efecto si la aplicación receptora no ha establecido una opción MQGMO_PROPERTIES o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.
V6COMPAT	Utilice esta opción para recibir MQRFH2 en el mismo formato que se envió. Si la aplicación emisora o el gestor de colas crea propiedades de mensaje adicionales, se devuelven en el manejador de mensajes. Esta opción se debe establecer en las colas emisora y receptora y en cualquier cola de transmisión que intervenga. Prevalece sobre cualquier opción PROPCTL establecido en definiciones de cola en la vía de acceso de resolución de nombres de colas. Utilice la opción V6COMPAT sólo en circunstancias excepcionales. Por ejemplo, si está migrando aplicaciones de una versión anterior a IBM MQ 9.1, la opción es valiosa porque conserva el comportamiento de la versión anterior. La opción es probable que afecte al rendimiento de los mensajes. También es más difícil de administrar; deberá asegurarse de que la opción esté establecida en las colas emisora, receptora e interviniente. V6COMPAT sólo tiene este efecto si la aplicación receptora no ha establecido una opción MQGMO_PROPERTIES o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.

Para obtener más información sobre las propiedades de mensaje y los pares de nombre-valor, consulte “Datos de NameValue(MQCHARn)” en la página 539.

[1] La existencia de carpetas de propiedades específicas creadas por IBM MQ classes for JMS indica un mensaje JMS . Las carpetas de propiedades son mcd., jms., usr.o mqext.

Opciones de propiedad de mensaje para MQGMO

Utilice las opciones de propiedad de mensaje **MQGMO** para controlar cómo se devuelven las propiedades de mensaje a una aplicación.

Tabla 493. Valores de las opciones de propiedades de mensaje MQGMO

MQGMO Option	**Descripción**
MQGMO_PROPERTIES_AS_Q_DEF	Las aplicaciones IBM MQ que leen de la misma cola y no establecen GMO_PROPERTIES_*, reciben las propiedades de mensaje de forma diferente. Las aplicaciones IBM MQ que no crean un manejador de mensajes, se controlan mediante el atributo **PROPCTL** de la cola. Una aplicación IBM MQ puede elegir recibir propiedades de mensaje en MQRFH2, o crear un descriptor de contexto de mensaje y consultar las propiedades de mensaje. Si la aplicación crea un manejador de mensajes, las propiedades se eliminan de MQRFH2. • Una aplicación IBM MQ nueva o cambiada que no define GMO_PROPERTIES_* o que establece su valor en MQGMO_PROPERTIES_AS_Q_DEF puede elegir consultar propiedades de mensaje. Debe definir MQCRTMH para crear un manejador de mensajes y consultar propiedades de mensaje utilizando la llamada de MQI MQINQMP. • Si una aplicación nueva o modificada no crea un descriptor de mensaje, debe leer las propiedades de mensaje que recibe directamente de las cabeceras MQRFH2 . • Si el atributo de cola **PROPCTL** se establece en FORCE, no se devuelven propiedades en el manejador de mensajes. Todas las propiedades se devuelven en cabeceras MQRFH2. • Si el atributo de cola **PROPCTL** se establece en NONE, o COMPAT, una aplicación IBM MQ que crea un descriptor de contexto de mensaje, recibe todas las propiedades de mensaje.
MQGMO_PROPERTIES_IN_HANDLE	Fuerza una aplicación a utilizar propiedades de mensaje. Utilice esta opción para detectar si una aplicación modificada no logra crear un manejador de mensajes. La aplicación podría estar intentando leer propiedades de mensaje directamente de una MQRFH2, en lugar de llamar a MQINQMP.
MQGMO_NO_PROPERTIES	• Se eliminan todas las propiedades. Las propiedades generadas por el gestor de colas como, por ejemplo, propiedades JMS, se eliminan. • Las propiedades se eliminan aunque se cree un manejador de mensajes. Los pares de nombre-valor de otras carpetas MQRFH2 están disponibles en los datos del mensaje.
MQGMO_PROPERTIES_FORCE_MQRFH2	Las propiedades se devuelven en las cabeceras MQRFH2, aunque se cree un manejador de mensajes. • MQINQMP no devuelve ninguna propiedad de mensaje, aunque se cree un manejador de mensajes. Se devuelve MQRC_PROPERTY_NOT_AVAILABLE si se consulta una propiedad.

Tabla 493. Valores de las opciones de propiedades de mensaje MQGMO (continuación)	
MQGMO Option	**Descripción**
MQGMO_PROPERTIES_COMPATIBILITY	Si el mensaje proviene de un cliente JMS, las propiedades JMS se devuelven en las cabeceras MQRFH2. Las aplicaciones IBM MQ nuevas o modificadas que crean un manejador de mensajes se comportan de forma diferente. • Todas las propiedades en cualquier carpeta de propiedades de mensaje se devuelven si el mensaje contiene una carpeta mcd., jms., usr.o mqext. • Si el mensaje contiene carpetas de propiedad, pero no una carpeta mcd., jms., usr.o mqext, no se devolverá ninguna propiedad de mensaje en una MQRFH2. • Si una aplicación nueva o modificada de la IBM MQ crea un descriptor de contexto de mensajes, consulte las propiedades de mensajes utilizando la llamada MQI MQINQMP. Todas las propiedades de mensaje se eliminan de MQRFH2. • Si una aplicación IBM MQ nueva o modificada crea un manejador de mensajes, se pueden consultar todas las propiedades contenidas en el mensaje. Incluso si el mensaje no contiene una carpeta mcd., jms., usr.o mqext, se pueden consultar todas las propiedades de mensaje.

Referencia relacionada
PROPCTL
2471 (09A7) (RC2471): MQRC_PROPERTY_NOT_AVAILABLE

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQGMO_STRUC_ID
Identificador de la estructura de opciones de obtención de mensaje.

Para el lenguaje de programación C, también se define la constante MQGMO_STRUC_ID_ARRAY; tiene el mismo valor que MQGMO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQGMO_STRUC_ID.

Versión (MQLONG)

La versión es el número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQGMO_VERSION_1
Estructura de opciones de obtención de mensaje Version-1 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_2
Estructura de opciones de obtención de mensaje Version-2 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_3
Estructura de opciones de obtención de Version-3 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_4
Estructura de opciones de obtención de mensaje Version-4 .

Esta versión está soportada en todos los entornos.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQGMO_CURRENT_VERSION
Versión actual de la estructura de opciones de obtención de mensajes.

Siempre es un campo de entrada. El valor inicial de este campo es MQGMO_VERSION_1.

Opciones (MQLONG) para MQGMO

Las opciones de **MQGMO** controlan la acción de MQGET. Puede especificar cero o más de las opciones. Si necesita más de un valor opcional:

- Añada los valores (no añada la misma constante más de una vez), o
- Combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

Las combinaciones de opciones que no son válidas se anotan; todas las demás combinaciones son válidas.

Opciones de espera

Las opciones siguientes están relacionadas con la espera de que lleguen mensajes a la cola:

MQGMO_WAIT
La aplicación espera hasta que llega un mensaje adecuado. El tiempo máximo que la aplicación espera se especifica en *WaitInterval* .

Importante: No hay espera, ni retardo, si un mensaje adecuado está disponible inmediatamente.

Si se inhiben las solicitudes MQGET o las solicitudes MQGET se inhiben mientras se espera, se cancela la espera. La llamada se completa con MQCC_FAILED y el código de razón MQRC_GET_INHIBITED, independientemente de si hay mensajes adecuados en la cola.

Puede utilizar MQGMO_WAIT con las opciones MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT .

Si varias aplicaciones están esperando en la misma cola compartida, las reglas siguientes seleccionan qué aplicación se activa cuando llega un mensaje adecuado:

Tabla 494. Reglas para activar llamadas de MQGET en una cola compartida.

Número de llamadas de MQGET en espera de ser activadas		**Resultado**
Con una opción BROWSE	**Sin una opción BROWSE**[2]	
Ninguna	uno o más	Se activa una llamada MQGET sin una opción BROWSE .
uno o más	Ninguna	Todas las llamadas de MQGET con una opción BROWSE están activadas.
uno o más	uno o más	Se activa una llamada MQGET sin una opción BROWSE . El número de llamadas MQGET con una opción BROWSE que están activadas es imprevisible.

Si más de una llamada MQGET sin una opción BROWSE está esperando en la misma cola, sólo se activa una. El gestor de colas intenta dar prioridad a las llamadas en espera en el orden siguiente:

[2] Una llamada MQGET que especifica la opción MQGMO_LOCK se trata como una llamada de no examen.

1. Solicitudes get-wait específicas que sólo pueden satisfacerse mediante determinados mensajes, por ejemplo, aquellos con un `MsgId` o `CorrelId` específico (o ambos).
2. Solicitudes de obtención de espera generales que pueden ser satisfechas por cualquier mensaje.

Nota:

- Dentro de la primera categoría, no se otorga ninguna prioridad adicional a las solicitudes get-wait más específicas. Por ejemplo, las solicitudes que especifican `MsgId` y `CorrelId` .
- Dentro de cualquiera de las dos categorías, no se puede predecir qué aplicación está seleccionada. En concreto, la aplicación que más espera no es necesariamente la seleccionada.
- La longitud de vía de acceso y las consideraciones de planificación de prioridad del sistema operativo pueden significar que una aplicación en espera de una prioridad de sistema operativo inferior a la esperada recupera el mensaje.
- También puede suceder que una aplicación que no está en espera recupere el mensaje en lugar de uno que sí lo está.

z/OS En z/OS, se aplican los puntos siguientes:

- Si desea que la aplicación continúe con otro trabajo mientras espera a que llegue el mensaje, considere la posibilidad de utilizar la opción de señal (`MQGMO_SET_SIGNAL`) en su lugar. Sin embargo, la opción de señal es específica del entorno; las aplicaciones que se van a portar entre distintos entornos no deben utilizarla.
- Si hay más de una llamada `MQGET` en espera del mismo mensaje, con una combinación de opciones de espera y de señal, cada llamada en espera se considera igualmente. Es un error especificar `MQGMO_SET_SIGNAL` con `MQGMO_WAIT`. También es un error especificar esta opción con un descriptor de contexto de cola para el que hay una señal pendiente.
- Si especifica `MQGMO_WAIT` o `MQGMO_SET_SIGNAL` para una cola que tiene un `IndexType` de `MQIT_MSG_TOKEN`, no se permiten criterios de selección. Esto significa que:
  - Si está utilizando una version-1 `MQGMO`, establezca los campos `MsgId` y `CorrelId` en el `MQMD` especificado en la llamada `MQGET` en `MQMI_NONE` y `MQCI_NONE`.
  - Si está utilizando una version-2 o posterior `MQGMO`, establezca el campo `MatchOptions` en `MQMO_NONE`.
- Para una llamada `MQGET` en una cola compartida y la llamada es una solicitud de examen, o una obtención destructiva de un mensaje de grupo, y ni `MsgId` ni `CorrelId` deben coincidir, la señal ECB se publica MQEC_MSG_LLEGÓ después de 200 milisegundos.

  Esto ocurre, aunque es posible que no haya llegado un mensaje adecuado a la cola, hasta que haya caducado el intervalo de espera, cuando la cola se envía con MQEC_WAIT_INTERVAL_EXPIRED. Cuando se publica MQEC_MSG_RECIBIDO, debe volver a emitir una segunda llamada `MQGET` para recuperar el mensaje, si hay uno disponible.

  Esta técnica se utiliza para asegurarse de que se le informa a tiempo de la llegada de un mensaje, pero puede aparecer como una sobrecarga de proceso inesperada cuando se compara con una secuencia de llamada similar en una cola no compartida.

`MQGMO_WAIT` se ignora si se especifica con `MQGMO_BROWSE_MSG_UNDER_CURSOR` o `MQGMO_MSG_UNDER_CURSOR` ; no se genera ningún error.

MQGMO_NO_WAIT
La aplicación no espera si no hay ningún mensaje adecuado disponible. `MQGMO_NO_WAIT` es lo opuesto a `MQGMO_WAIT`. `MQGMO_NO_WAIT` está definido para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

MQGMO_SET_SIGNAL
Utilice esta opción con los campos `Signal1` y `Signal2` . Permite a las aplicaciones continuar con otro trabajo mientras esperan a que llegue un mensaje. También permite a las aplicaciones (si hay disponibles recursos adecuados del sistema operativo) esperar a que lleguen mensajes a más de una cola.

Nota: La opción MQGMO_SET_SIGNAL es específica del entorno; no la utilice para las aplicaciones que desea portar.

En dos circunstancias, la llamada se completa de la misma forma que si no se hubiera especificado esta opción:

1. Si un mensaje disponible actualmente cumple los criterios especificados en el descriptor de mensaje.
2. Si se detecta un error de parámetro u otro error síncrono.

Si no hay ningún mensaje que cumpla los criterios especificados en el descriptor de mensaje disponible actualmente, el control vuelve a la aplicación sin esperar a que llegue un mensaje. Los parámetros **CompCode** y **Reason** se establecen en MQCC_WARNING y MQRC_SIGNAL_REQUEST_ACCEPTED. No se han establecido otros campos de salida en el descriptor de mensaje y los parámetros de salida de la llamada MQGET . Cuando un mensaje adecuado llega más tarde, la señal se emite mediante la publicación del BCE.

A continuación, el llamante debe volver a emitir la llamada MQGET para recuperar el mensaje. La aplicación puede esperar esta señal, utilizando funciones proporcionadas por el sistema operativo.

Si el sistema operativo proporciona un mecanismo de espera múltiple, puede utilizarlo para esperar a que llegue un mensaje en cualquiera de varias colas.

Si se especifica un WaitInterval distinto de cero, la señal se entrega después de que caduque el intervalo de espera. El gestor de colas también puede cancelar la espera, en cuyo caso se entrega la señal.

Más de una llamada MQGET puede establecer una señal para el mismo mensaje. El orden en el que se activan las aplicaciones es el mismo que el descrito para MQGMO_WAIT.

Si más de una llamada MQGET está esperando el mismo mensaje, cada llamada en espera se considera igualmente. Las llamadas pueden incluir una mezcla de opciones de espera y de señal.

Bajo determinadas condiciones, la llamada MQGET puede recuperar un mensaje y se puede entregar una señal resultante de la llegada del mismo mensaje. Cuando se entrega una señal, una aplicación debe estar preparada para que no haya ningún mensaje disponible.

Un descriptor de contexto de cola no puede tener más de una solicitud de señal pendiente.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_UNLOCK
- MQGMO_WAIT

Para una llamada MQGET en una cola compartida y la llamada es una solicitud de examen, o una obtención destructiva de un mensaje de grupo, y ni MsgId ni CorrelId deben coincidir, la señal del usuario ECB se publica MQEC_MSG_ARRIVED después de 200 milisegundos.

Esto ocurre, aunque es posible que no haya llegado un mensaje adecuado a la cola, hasta que haya caducado el intervalo de espera, cuando la cola se envía con MQEC_WAIT_INTERVAL_EXPIRED. Cuando se publica MQEC_MSG_ARRIVED , debe volver a emitir una segunda llamada MQGET para recuperar el mensaje, si hay una disponible.

Esta técnica se utiliza para asegurarse de que se le informa a tiempo de la llegada de un mensaje, pero puede aparecer como una sobrecarga de proceso inesperada cuando se compara con una secuencia de llamada similar en una cola no compartida.

No es un método eficaz de recuperación de mensajes cuando los mensajes se añaden con poca frecuencia. Para evitar esta sobrecarga para el caso de examen, especifique MsgId (si no está indexado o indexado por MsgId) o *CorrelId* (si está indexado por CorrelId) coincidente en la llamada MQGET .

z/OS Esta opción solo está soportada en z/OS .

MQGMO_FAIL_IF_QUIESCING

Fuerza a que la llamada MQGET falle si el gestor de colas está en estado de desactivación temporal.

z/OS En z/OS, esta opción también fuerza que la llamada MQGET falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

Si esta opción se especifica con MQGMO_WAIT o MQGMO_SET_SIGNAL, y la espera o señal está pendiente en el momento en que el gestor de colas entra en el estado de desactivación temporal:

- La espera se cancela y la llamada devuelve el código de terminación MQCC_FAILED con el código de razón MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.
- La señal se cancela con un código de terminación de señal específico del entorno.

  z/OS En z/OS, la señal se completa con el código de terminación de suceso MQEC_Q_MGR_QUIESCING o MQEC_CONNECTION_QUIESCING.

Si no se especifica MQGMO_FAIL_IF_QUIESCING y el gestor de colas o la conexión entra en el estado de desactivación temporal, la espera o la señal no se cancela.

Opciones de punto de sincronización

Las siguientes opciones están relacionadas con la participación de la llamada MQGET dentro de una unidad de trabajo:

MQGMO_SYNCPOINT

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

Puede dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT sin establecer. En este caso, la inclusión de la solicitud de obtención en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas. No lo determina el entorno que ejecuta la aplicación.

- z/OS En z/OS, la solicitud de obtención está dentro de una unidad de trabajo.
- En todos los entornos excepto z/OS, la solicitud de obtención no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_SYNCPOINT_IF_PERSISTENT

La solicitud es operar dentro de los protocolos de unidad de trabajo normales, pero sólo si el mensaje recuperado es persistente. Un mensaje persistente tiene el valor MQPER_PERSISTENT en el campo Persistence en MQMD.

- Si el mensaje es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_SYNCPOINT.
- Si el mensaje no es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_NO_SYNCPOINT.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_COMPLETE_MSG
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT
- MQGMO_UNLOCK

Esta opción está soportada en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

MQGMO_NO_SYNCPOINT

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. Si obtiene un mensaje sin una opción de examen, se suprime de la cola inmediatamente. El mensaje no puede volver a estar disponible restituir la unidad de trabajo.

Esta opción se presupone si especifica MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Puede dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT sin establecer. En este caso, la inclusión de la solicitud de obtención en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas. No lo determina el entorno que ejecuta la aplicación.

- z/OS En z/OS, la solicitud de obtención está dentro de una unidad de trabajo.
- En todos los entornos excepto z/OS, la solicitud de obtención no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

z/OS **MQGMO_MARK_SKIP_BACKOUT**

Restituir una unidad de trabajo sin restablecer en la cola el mensaje que se ha marcado con esta opción.

Esta opción sólo está soportada en z/OS.

Si se especifica esta opción, también se debe especificar MQGMO_SYNCPOINT . MQGMO_MARK_SKIP_BACKOUT no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Nota: En IMS y CICS, es posible que tenga que emitir una llamada extran IBM MQ después de restituir una unidad de trabajo que contiene un mensaje marcado con MQGMO_MARK_SKIP_BACKOUT. Debe emitir una llamada IBM MQ antes de confirmar la nueva unidad de trabajo que contiene el mensaje marcado. La llamada puede ser cualquier llamada de IBM MQ que desee.

1. En IMS, si no ha aplicado IMS APAR PN60855 y está ejecutando una aplicación IMS MPP o BMP.
2. En CICS, si está ejecutando alguna aplicación.

En ambos casos, emita cualquier llamada IBM MQ antes de confirmar la nueva unidad de trabajo que contiene el mensaje restituido.

Nota: Dentro de una unidad de trabajo, sólo puede haber una solicitud de obtención marcada como omisión de restitución, así como ninguna o varias solicitudes de obtención no marcadas.

Si una aplicación se restituye de una unidad de trabajo, un mensaje que se ha recuperado utilizando MQGMO_MARK_SKIP_BACKOUT no se restaura a su estado anterior. Otras actualizaciones de recursos se restituyen. El mensaje se trata como si se hubiera recuperado en una nueva unidad de trabajo iniciada por la solicitud de restitución. El mensaje se recupera sin la opción MQGMO_MARK_SKIP_BACKOUT .

MQGMO_MARK_SKIP_BACKOUT es útil si, después de que se hayan cambiado algunos recursos, se hace evidente que la unidad de trabajo no se puede completar correctamente. Si omite esta opción, restituir la unidad de trabajo restablece el mensaje en la cola. Se vuelve a producir la misma secuencia de sucesos, cuando el mensaje se recupera a continuación.

Sin embargo, si especifica MQGMO_MARK_SKIP_BACKOUT en la llamada MQGET original, restituir la unidad de trabajo restituye las actualizaciones en los otros recursos. El mensaje se trata como si se hubiera recuperado bajo una nueva unidad de trabajo. La aplicación puede realizar el manejo de errores adecuado. Puede enviar un mensaje de informe al remitente del mensaje original o colocar el mensaje original en la cola de mensajes no entregados. A continuación, puede confirmar la nueva unidad de trabajo. La confirmación de la nueva unidad de trabajo elimina el mensaje de forma permanente de la cola original.

MQGMO_MARK_SKIP_BACKOUT marca un único mensaje físico. Si el mensaje pertenece a un grupo de mensajes, los demás mensajes del grupo no se marcan. De forma similar, si el mensaje marcado es un segmento de un mensaje lógico, los demás segmentos del mensaje lógico no se marcan.

Cualquier mensaje de un grupo se puede marcar, pero si los mensajes se recuperan utilizando MQGMO_LOGICAL_ORDER, es conveniente marcar el primer mensaje del grupo. Si la unidad de trabajo se restituye, el primer mensaje (marcado) se mueve a la nueva unidad de trabajo. El segundo y los mensajes posteriores del grupo se restablecen en la cola. Los mensajes que quedan en la cola no pueden ser recuperados por otra aplicación utilizando MQGMO_LOGICAL_ORDER. El primer mensaje del grupo ya no está en la cola. Sin embargo, la aplicación que ha realizado la copia de seguridad de la unidad de trabajo puede recuperar el segundo y los mensajes posteriores en la nueva unidad de trabajo utilizando la opción MQGMO_LOGICAL_ORDER . El primer mensaje ya se ha recuperado.

Ocasionalmente es posible que tenga que restituir la nueva unidad de trabajo. Por ejemplo, porque la cola de mensajes no entregados está llena y el mensaje no debe descartarse. Al restituir la nueva unidad de trabajo se restablece el mensaje en la cola original, lo que impide que se pierda el mensaje. Sin embargo, en esta situación el proceso no puede continuar. Después de restituir la nueva unidad de trabajo, la aplicación debe informar al operador o administrador de que hay un error irrecuperable y, a continuación, finalizar.

MQGMO_MARK_SKIP_BACKOUT sólo funciona si la unidad de trabajo que contiene la solicitud de obtención se interrumpe cuando la aplicación la restituya. Si la unidad de trabajo que contiene la solicitud de obtención se restituye porque la transacción o el sistema falla, se ignora

MQGMO_MARK_SKIP_BACKOUT . Cualquier mensaje recuperado utilizando esta opción se restablece en la cola de la misma forma que los mensajes recuperados sin esta opción.

Examinar opciones

Las opciones siguientes están relacionadas con el examen de mensajes en la cola:

MQGMO_BROWSE_FIRST

Cuando se abre una cola con la opción MQOO_BROWSE, se establece un cursor para examinar, situado lógicamente antes del primer mensaje de la cola. A continuación, puede utilizar las llamadas MQGET especificando la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXTo MQGMO_BROWSE_MSG_UNDER_CURSOR para recuperar mensajes de la cola de forma no destructiva. El cursor para examinar marca la posición, dentro de los mensajes de la cola, desde la que la siguiente llamada MQGET con MQGMO_BROWSE_NEXT busca un mensaje adecuado.

MQGMO_BROWSE_FIRST no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

Una llamada MQGET con MQGMO_BROWSE_FIRST ignora la posición anterior del cursor para examinar. Se recupera el primer mensaje de la cola que cumple las condiciones especificadas en el descriptor de mensaje. El mensaje permanece en la cola y el cursor para examinar se coloca en este mensaje.

Después de esta llamada, el cursor para examinar se coloca en el mensaje que se ha devuelto. Es posible que el mensaje se elimine de la cola antes de que se emita la siguiente llamada MQGET con MQGMO_BROWSE_NEXT . En este caso, el cursor para examinar permanece en la posición de la cola que ocupaba el mensaje, aunque esa posición esté ahora vacía.

Utilice la opción MQGMO_MSG_UNDER_CURSOR con una llamada MQGET que no sea examinar, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET de no examinar, aunque se utilice el mismo descriptor de contexto *Hobj* . Tampoco se mueve mediante una llamada MQGET de examen que devuelve un código de terminación de MQCC_FAILED, o un código de razón de MQRC_TRUNCATED_MSG_FAILED.

Especifique la opción MQGMO_LOCK con esta opción, para bloquear el mensaje que se examina.

Puede especificar MQGMO_BROWSE_FIRST con cualquier combinación válida de las opciones MQGMO_* y MQMO_* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esta opción, los mensajes se examinan en orden físico. Si especifica MQGMO_BROWSE_FIRST, puede conmutar entre el orden lógico y el orden físico. Las llamadas MQGET subsiguientes que utilizan MQGMO_BROWSE_NEXT examinan la cola en el mismo orden que la llamada más reciente que ha especificado MQGMO_BROWSE_FIRST para el descriptor de contexto de cola.

El gestor de colas retiene dos conjuntos de información de grupo y segmento para llamadas MQGET . La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola. Si especifica MQGMO_BROWSE_FIRST, el gestor de colas ignora la información de grupo y segmento para examinar. Explora la cola como si no hubiera ningún grupo actual y ningún mensaje lógico actual. Si la llamada MQGET es satisfactoria, código de terminación MQCC_OK o MQCC_WARNING, la información de grupo y

segmento para examinar se establece en la del mensaje devuelto. Si la llamada falla, la información de grupo y segmento permanece igual que antes de la llamada.

MQGMO_BROWSE_NEXT
Avance el cursor de examen hasta el siguiente mensaje de la cola que cumpla los criterios de selección especificados en la llamada MQGET . El mensaje se devuelve a la aplicación, pero permanece en la cola.

MQGMO_BROWSE_NEXT no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

MQGMO_BROWSE_NEXT se comporta de la misma forma que MQGMO_BROWSE_FIRST, si es la primera llamada para examinar una cola, después de que se haya abierto la cola para examinarla.

Es posible que el mensaje bajo el cursor se elimine de la cola antes de que se emita la siguiente llamada MQGET con MQGMO_BROWSE_NEXT . El cursor para examinar permanece lógicamente en la posición de la cola que ocupaba el mensaje, aunque dicha posición esté ahora vacía.

Los mensajes se almacenan en la cola de una de estas dos maneras:

- FIFO dentro de prioridad (MQMDS_PRIORITY), o
- FIFO independientemente de la prioridad (MQMDS_FIFO)

El atributo de cola **MsgDeliverySequence** indica qué método se aplica (consulte “Atributos para colas” en la página 848 para obtener más detalles).

Una cola puede tener un MsgDeliverySequence de MQMDS_PRIORITY. Llega un mensaje a la cola que tiene una prioridad más alta que la a la que apunta actualmente el cursor para examinar. En cuyo caso, el mensaje de prioridad más alta no se encuentra durante el barrido actual de la cola utilizando MQGMO_BROWSE_NEXT. Sólo se puede encontrar después de que el cursor para examinar se haya restablecido con MQGMO_BROWSE_FIRST, o volviendo a abrir la cola.

La opción MQGMO_MSG_UNDER_CURSOR se puede utilizar con una llamada MQGET sin examinar si es necesario, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante llamadas MQGET no examinar utilizando el mismo descriptor de contexto Hobj .

Especifique la opción MQGMO_LOCK con esta opción para bloquear el mensaje que se examina.

Puede especificar MQGMO_BROWSE_NEXT con cualquier combinación válida de las opciones MQGMO_* y MQMO_* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esta opción, los mensajes se examinan en orden físico. Si especifica MQGMO_BROWSE_FIRST, puede conmutar entre el orden lógico y el orden físico. Las llamadas MQGET subsiguientes que utilizan MQGMO_BROWSE_NEXT examinan la cola en el mismo orden que la llamada más reciente que ha especificado MQGMO_BROWSE_FIRST para el descriptor de contexto de cola. La llamada falla con el código de razón MQRC_INCONSISTENT_BROWSE si no se cumple esta condición.

Nota: Tenga especial cuidado al utilizar una llamada MQGET para examinar más allá del final de un grupo de mensajes si no se especifica MQGMO_LOGICAL_ORDER . Por ejemplo, supongamos que el último mensaje del grupo precede al primer mensaje del grupo de la cola. Si se utiliza MQGMO_BROWSE_NEXT para examinar más allá del final del grupo, si se especifica

MQMO_MATCH_MSG_SEQ_NUMBER con MsgSeqNumber establecido en 1 , se devuelve el primer mensaje del grupo ya examinado. Este resultado puede producirse inmediatamente, o una serie de llamadas de MQGET más tarde si hay grupos intervinientes. La misma consideración se aplica a un mensaje lógico que no está en un grupo.

La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola.

MQGMO_BROWSE_MSG_UNDER_CURSOR

Recupere el mensaje al que apunta el cursor para examinar de forma no destructiva, independientemente de las opciones MQMO_* especificadas en el campo MatchOptions en MQGMO.

MQGMO_BROWSE_MSG_UNDER_CURSOR no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

El mensaje al que apunta el cursor para examinar es el último que se ha recuperado utilizando la opción MQGMO_BROWSE_FIRST o la opción MQGMO_BROWSE_NEXT . La llamada falla si no se ha emitido ninguna de estas llamadas para esta cola desde que se abrió. La llamada también falla si el mensaje que estaba bajo el cursor para examinar se ha recuperado de forma destructiva.

Esta llamada no cambia la posición del cursor para examinar.

La opción MQGMO_MSG_UNDER_CURSOR se puede utilizar con una llamada MQGET sin examinar, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET de no examinar, aunque se utilice el mismo descriptor de contexto Hobj . Tampoco se mueve mediante una llamada MQGET de examen que devuelve un código de terminación de MQCC_FAILED, o un código de razón de MQRC_TRUNCATED_MSG_FAILED.

Si se especifica MQGMO_BROWSE_MSG_UNDER_CURSOR con MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor, de modo que se devuelva sin desbloquear ni bloquear de nuevo. El mensaje permanece bloqueado.
- Si no hay ningún mensaje bloqueado y hay un mensaje bajo el cursor para examinar, se bloquea y se devuelve a la aplicación. Si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica MQGMO_BROWSE_MSG_UNDER_CURSOR sin MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor. El mensaje se devuelve a la aplicación y, a continuación, se desbloquea. Puesto que el mensaje está ahora desbloqueado, no hay garantía de que la misma aplicación pueda volver a examinarlo o recuperarlo de forma destructiva. Es posible que otra aplicación lo haya recuperado de forma destructiva obteniendo mensajes de la cola.
- Si no hay ningún mensaje bloqueado y hay un mensaje bajo el cursor para examinar, se devuelve a la aplicación. Si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_BROWSE_MSG_UNDER_CURSOR, el cursor para examinar debe identificar un mensaje cuyo campo Offset en MQMD sea cero. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola.

MQGMO_MSG_UNDER_CURSOR

Recupere el mensaje al que apunta el cursor para examinar, independientemente de las opciones MQMO_* especificadas en el campo MatchOptions en MQGMO. El mensaje se elimina de la cola.

El mensaje al que apunta el cursor para examinar es el último que se ha recuperado utilizando la opción MQGMO_BROWSE_FIRST o la opción MQGMO_BROWSE_NEXT .

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_MSG_UNDER_CURSOR, el cursor para examinar debe identificar un mensaje cuyo campo Offset en MQMD sea cero. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto tanto para examinar como para entrada. Si el cursor para examinar no apunta actualmente a un mensaje recuperable, la llamada MQGET devuelve un error.

MQGMO_MARK_BROWSE_HANDLE

El mensaje devuelto por un MQGETsatisfactorio, o identificado por el MsgTokendevuelto, está marcado. La marca es específica del descriptor de objeto utilizado en la llamada.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_HANDLE sólo es válido si también se especifica una de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE no es válido con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

El mensaje permanece en este estado hasta que se produce uno de los sucesos siguientes:

- El descriptor de contexto de objeto en cuestión está cerrado, ya sea normalmente o de otro modo.
- El mensaje no está marcado para este descriptor de contexto mediante una llamada a MQGET con la opción MQGMO_UNMARK_BROWSE_HANDLE.
- El mensaje se devuelve de una llamada a MQGETdestructivo, que se completa con MQCC_OK o MQCC_WARNING. El estado del mensaje permanece modificado incluso si el MQGET se retrotrae más tarde.
- El mensaje caduca.

MQGMO_MARK_BROWSE_CO_OP

El mensaje devuelto por un MQGETsatisfactorio, o identificado por el *MsgToken*devuelto, se marca para todos los descriptores de contexto del conjunto cooperante.

La marca de nivel de cooperación se suma a cualquier marca de nivel de descriptor de contexto que se pueda haber establecido.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_CO_OP sólo es válido si el descriptor de objeto utilizado ha sido devuelto por una llamada a MQOPEN que ha especificado MQOO_CO_OP. También debe especificar una de las siguientes opciones MQGMO :

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Si el mensaje ya está marcado y no se ha especificado la opción MQGMO_UNMARKED_BROWSE_MSG , la llamada falla con MQCC_FAILED y el código de razón MQRC_MSG_MARKED_BROWSE_CO_OP.

El mensaje permanece en este estado hasta que se produce uno de los sucesos siguientes:

- Se cierran todos los manejadores de objetos del conjunto cooperante.
- El mensaje no está marcado para los navegadores que cooperan mediante una llamada a MQGET con la opción MQGMO_UNMARK_BROWSE_CO_OP.
- El gestor de colas desmarca automáticamente el mensaje.
- El mensaje se devuelve desde una llamada a un MQGETque no es examinar. El estado del mensaje permanece modificado incluso si el MQGET se retrotrae más tarde.
- El mensaje caduca.

MQGMO_UNMARKED_BROWSE_MSG

Una llamada a MQGET que especifica MQGMO_UNMARKED_BROWSE_MSG devuelve un mensaje que se considera no marcado para su descriptor de contexto. No devuelve un mensaje si el mensaje se ha marcado para su descriptor de contexto. Tampoco devuelve el mensaje si la cola se ha abierto mediante una llamada a MQOPEN, con la opción MQOO_CO_OP, y el mensaje ha sido marcado por un miembro del conjunto cooperante.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP

Después de una llamada a MQGET que especifica esta opción, los descriptores de contexto abiertos del conjunto de descriptores de contexto cooperantes ya no consideran que el mensaje se marque para el conjunto cooperante. El mensaje se sigue considerando marcado a nivel de descriptor de contexto si se ha marcado a nivel de descriptor de contexto antes de esta llamada.

El uso de MQGMO_UNMARK_BROWSE_CO_OP sólo es válido con un descriptor de contexto devuelto por una llamada satisfactoria a MQOPEN con la opción MQOO_CO_OP. MQGET se ejecuta correctamente aunque el mensaje no se considere marcado por el conjunto de descriptores de contexto cooperantes.

MQGMO_UNMARK_BROWSE_CO_OP no es válido en una llamada MQGET sin examinar, o con cualquiera de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

MQGMO_UNMARK_BROWSE_HANDLE

Después de una llamada a MQGET que especifica esta opción, el mensaje ubicado ya no se considera marcado por este descriptor de contexto.

La llamada se realiza correctamente incluso si el mensaje no está marcado para este descriptor de contexto.

Esta opción no es válida en una llamada MQGET sin examinar, o con cualquiera de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

Opciones de bloqueo

Las opciones siguientes están relacionadas con el bloqueo de mensajes en la cola:

MQGMO_LOCK

Bloquee el mensaje que se examina, de modo que el mensaje se vuelva invisible para cualquier otro descriptor de contexto abierto para la cola. La opción sólo se puede especificar si también se especifica una de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Sólo se puede bloquear un mensaje para cada descriptor de contexto de cola. El mensaje puede ser un mensaje lógico o un mensaje físico:

- Si especifica MQGMO_COMPLETE_MSG, todos los segmentos de mensaje que componen el mensaje lógico se bloquean en el descriptor de contexto de cola. Todos los mensajes deben estar presentes en la cola y disponibles para su recuperación.
- Si omite MQGMO_COMPLETE_MSG, sólo se bloquea un único mensaje físico en el descriptor de contexto de cola. Si este mensaje es un segmento de un mensaje lógico, el segmento bloqueado impide que otras aplicaciones utilicen MQGMO_COMPLETE_MSG para recuperar o examinar el mensaje lógico.

El mensaje bloqueado es siempre el que está bajo el cursor para examinar. El mensaje se puede eliminar de la cola mediante una llamada MQGET posterior que especifique la opción

MQGMO_MSG_UNDER_CURSOR . Otras llamadas de MQGET que utilizan el descriptor de contexto de cola también pueden eliminar el mensaje (por ejemplo, una llamada que especifica el identificador de mensaje del mensaje bloqueado).

Si la llamada devuelve el código de terminación MQCC_FAILEDo MQCC_WARNING con el código de razón MQRC_TRUNCATED_MSG_FAILED, no se bloquea ningún mensaje.

Si la aplicación no elimina el mensaje de la cola, el bloqueo se libera mediante una de las acciones siguientes:

- Emitiendo otra llamada MQGET para este descriptor de contexto, especificando MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT. El bloqueo se libera si la llamada se completa con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada se completa con MQCC_FAILED. No obstante, se aplican las excepciones siguientes:
  - El mensaje no se desbloquea si se devuelve MQCC_WARNING con MQRC_TRUNCATED_MSG_FAILED.
  - El mensaje se desbloquea si se devuelve MQCC_FAILED con MQRC_NO_MSG_AVAILABLE.

  Si también especifica MQGMO_LOCK, el mensaje devuelto se bloquea. Si omite MQGMO_LOCK, no hay ningún mensaje bloqueado después de la llamada.

  Si especifica MQGMO_WAITy no hay ningún mensaje disponible inmediatamente, el mensaje original se desbloquea antes del inicio de la espera.
- Emitiendo otra llamada MQGET para este descriptor de contexto, con MQGMO_BROWSE_MSG_UNDER_CURSOR, sin MQGMO_LOCK. El bloqueo se libera si la llamada se completa con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada se completa con MQCC_FAILED. Sin embargo, se aplica la siguiente excepción:
  - El mensaje no se desbloquea si se devuelve MQCC_WARNING con MQRC_TRUNCATED_MSG_FAILED.
- Emitiendo otra llamada MQGET para este descriptor de contexto con MQGMO_UNLOCK.
- Emisión de una llamada MQCLOSE utilizando el descriptor de contexto. El MQCLOSE puede estar implícito, provocado por la finalización de la aplicación.

No es necesaria ninguna opción MQOPEN especial para especificar MQGMO_LOCK, que no sea MQOO_BROWSE, que es necesaria para especificar una opción de examen que la acompañe.

MQGMO_LOCK no es válido con ninguna de las opciones siguientes:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_UNLOCK

El mensaje que se va a desbloquear debe haber sido bloqueado previamente por una llamada MQGET con la opción MQGMO_LOCK . Si no hay ningún mensaje bloqueado para este descriptor de contexto, la llamada se completa con MQCC_WARNING y MQRC_NO_MSG_LOCKED.

Los parámetros **MsgDesc**, **BufferLength**, **Buffer**y **DataLength** no se comprueban ni modifican si especifica MQGMO_UNLOCK. No se devuelve ningún mensaje en *Buffer*.

No es necesaria ninguna opción de apertura especial para especificar MQGMO_UNLOCK (aunque se necesita MQOO_BROWSE para emitir la solicitud de bloqueo en primer lugar).

Esta opción no es válida con ninguna opción excepto la siguiente:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Ambas opciones se asumen tanto si se especifican como si no.

Opciones de datos de mensaje

Las opciones siguientes están relacionadas con el proceso de los datos de mensaje cuando el mensaje se lee de la cola:

MQGMO_ACCEPT_TRUNCATED_MSG

Si el almacenamiento intermedio de mensajes es demasiado pequeño para contener el mensaje completo, permita que la llamada MQGET llene el almacenamiento intermedio. MQGET llena el almacenamiento intermedio con la mayor parte del mensaje que puede. Emite un código de finalización de aviso y completa su proceso. Esto significa que:

- Al examinar mensajes, el cursor para examinar se avanza al mensaje devuelto.
- Al eliminar mensajes, el mensaje devuelto se elimina de la cola.
- Se devuelve el código de razón MQRC_TRUNCATED_MSG_ACCEPTED si no se produce ningún otro error.

Sin esta opción, el almacenamiento intermedio se sigue llenando con la mayor cantidad de mensajes que pueda contener. Se emite un código de finalización de aviso, pero el proceso no se ha completado. Esto significa que:

- Al examinar mensajes, el cursor para examinar no está avanzado.
- Al eliminar mensajes, el mensaje no se elimina de la cola.
- Se devuelve el código de razón MQRC_TRUNCATED_MSG_FAILED si no se produce ningún otro error.

MQGMO_CONVERT

Esta opción convierte los datos de aplicación en el mensaje para que se ajusten a los valores CodedCharSetId y Encoding especificados en el parámetro **MsgDesc** en la llamada MQGET . Los datos se convierten antes de que se copien en el parámetro **Buffer** .

El campo Format especificado cuando se colocó el mensaje es asumido por el proceso de conversión para identificar la naturaleza de los datos en el mensaje. El gestor de colas convierte los datos de mensaje para formatos incorporados y una salida escrita por el usuario para otros formatos. Consulte “salida de conversión de datos” en la página 923 para obtener más información sobre la salida de conversión de datos.

- Si la conversión es satisfactoria, los campos CodedCharSetId y Encoding especificados en el parámetro **MsgDesc** no se modifican al volver de la llamada MQGET .
- Si sólo falla la conversión, los datos del mensaje se devuelven sin convertir. Los campos CodedCharSetId y Encoding de MsgDesc se establecen en los valores del mensaje sin convertir. El código de terminación es MQCC_WARNING en este caso.

En cualquier caso, estos campos describen el identificador de juego de caracteres y la codificación de los datos de mensaje que se devuelven en el parámetro **Buffer** .

Consulte el campo *Format* descrito en “MQMD - Descriptor de mensaje” en la página 421 para obtener una lista de nombres de formato para los que el gestor de colas realiza la conversión.

Opciones de grupo y segmento

Las opciones siguientes están relacionadas con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Antes de las descripciones de las opciones, a continuación se muestran algunas definiciones de términos importantes:

Mensaje físico

Un mensaje físico es la unidad de información más pequeña que se puede colocar o eliminar de una cola. A menudo se corresponde con la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET . Cada mensaje físico tiene su propio descriptor de mensaje, MQMD. Normalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje, el campo MsgId en MQMD. El gestor de colas no impone valores diferentes.

Mensaje lógico

Un mensaje lógico es una única unidad de información de aplicación. En ausencia de restricciones del sistema, un mensaje lógico es el mismo que un mensaje físico. Si los mensajes lógicos son grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados segmentos.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo, `GroupId` campo en `MQMD`. Tienen el mismo número de secuencia de mensaje, `MsgSeqNumber` campo en `MQMD`. Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento, campo `Offset` en `MQMD`. El desplazamiento de segmento es el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Puesto que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo. En este caso, sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos, para los que la aplicación emisora ha inhibido la segmentación, tienen un identificador de grupo nulo, `MQGI_NONE`, a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por diferentes valores para el número de secuencia de mensaje. El número de secuencia es un entero comprendido entre 1 y n, donde n es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

MQGMO_LOGICAL_ORDER

`MQGMO_LOGICAL_ORDER` controla el orden en el que las llamadas `MQGET` sucesivas devuelven los mensajes para el descriptor de contexto de cola. La opción debe especificarse en cada llamada.

Si se especifica `MQGMO_LOGICAL_ORDER` para llamadas `MQGET` sucesivas para el mismo descriptor de contexto de cola, los mensajes de los grupos se devuelven en el orden de sus números de secuencia de mensajes. Los segmentos de mensajes lógicos se devuelven en el orden indicado por sus desplazamientos de segmento. Este orden puede ser diferente del orden en el que se producen estos mensajes y segmentos en la cola.

Nota: La especificación de `MQGMO_LOGICAL_ORDER` no tiene consecuencias adversas en los mensajes que no pertenecen a grupos y que no son segmentos. En efecto, estos mensajes se tratan como si cada uno perteneciera a un grupo de mensajes que consta de un solo mensaje. Es seguro especificar `MQGMO_LOGICAL_ORDER` cuando se recuperan mensajes de colas que contienen una mezcla de mensajes en grupos, segmentos de mensajes y mensajes no segmentados que no están en grupos.

Para devolver los mensajes en el orden necesario, el gestor de colas conserva la información de grupo y segmento entre llamadas `MQGET` sucesivas. La información de grupo y segmento identifica el grupo de mensajes actual y el mensaje lógico actual para el descriptor de contexto de cola. También identifica la posición actual dentro del grupo y el mensaje lógico, y si los mensajes se están recuperando dentro de una unidad de trabajo. Puesto que el gestor de colas conserva esta información, la aplicación no necesita establecer la información de grupo y segmento antes de cada llamada `MQGET` . Específicamente, significa que la aplicación no necesita establecer los campos `GroupId`, `MsgSeqNumbery` `Offset` en `MQMD`. Sin embargo, la aplicación debe establecer la opción `MQGMO_SYNCPOINT` o `MQGMO_NO_SYNCPOINT` correctamente en cada llamada.

Cuando se abre la cola, no hay ningún grupo de mensajes actual ni ningún mensaje lógico actual. Un grupo de mensajes se convierte en el grupo de mensajes actual cuando la llamada `MQGET` devuelve un mensaje que tiene el distintivo `MQMF_MSG_IN_GROUP` . Con `MQGMO_LOGICAL_ORDER` especificado en llamadas sucesivas, ese grupo sigue siendo el grupo actual hasta que se devuelve un mensaje que tiene:

- MQMF_LAST_MSG_IN_GROUP sin MQMF_SEGMENT (es decir, el último mensaje lógico del grupo no está segmentado), o
- MQMF_LAST_MSG_IN_GROUP con MQMF_LAST_SEGMENT (es decir, el mensaje devuelto es el último segmento del último mensaje lógico del grupo).

Cuando se devuelve un mensaje de este tipo, el grupo de mensajes termina y, al finalizar correctamente la llamada MQGET , ya no hay un grupo actual. De forma similar, un mensaje lógico se convierte en el mensaje lógico actual cuando la llamada MQGET devuelve un mensaje que tiene el distintivo MQMF_SEGMENT . El mensaje lógico termina cuando se devuelve el mensaje que tiene el distintivo MQMF_LAST_SEGMENT .

Si no se especifica ningún criterio de selección, las llamadas MQGET sucesivas devuelven, en el orden correcto, los mensajes para el primer grupo de mensajes de la cola. A continuación, devuelven los mensajes para el segundo grupo de mensajes, y así sucesivamente, hasta que no haya más mensajes disponibles. Es posible seleccionar los grupos de mensajes concretos devueltos especificando una o más de las opciones siguientes en el campo MatchOptions :

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Sin embargo, estas opciones sólo son efectivas cuando no hay ningún grupo de mensajes actual o mensaje lógico. Consulte el campo MatchOptions descrito en "MQGMO-Opciones de obtención de mensajes" en la página 367 para obtener más detalles.

La Tabla 495 en la página 392 muestra los valores de los campos MsgId, CorrelId, GroupId, MsgSeqNumbery Offset que el gestor de colas busca al intentar encontrar un mensaje para devolver en la llamada MQGET . Las reglas se aplican tanto a la eliminación de mensajes de la cola como a la exploración de mensajes en la cola. En la tabla, significa Sí o No:

LOG ORD
Indica si se ha especificado la opción MQGMO_LOGICAL_ORDER en la llamada.

Cur grp
Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg
Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas
Mostrar los valores que busca el gestor de colas. Anterior indica el valor devuelto para el campo en el mensaje anterior para el descriptor de contexto de cola.

Tabla 495. Opciones MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos

Opciones especificadas	**Estado de grupo y mensaje lógico antes de la llamada**		**Valores que el gestor de colas busca**				
LOG ORD	**Cur grp**	**Cur log msg**	*MsgId*	*CorrelId*	*GroupId*	*MsgSeqNumber*	*Offset*
Sí	No	No	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	1	0
Sí	No	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior

Tabla 495. Opciones MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos (continuación)							
Opciones especificadas	**Estado de grupo y mensaje lógico antes de la llamada**		**Valores que el gestor de colas busca**				
Sí	Sí	No	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	Cualquiera de los dos	Cualquiera de los dos	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*

Si hay varios grupos de mensajes en la cola y son elegibles para la devolución, los grupos se devuelven en el orden determinado por la posición en la cola del primer segmento del primer mensaje lógico de cada grupo. Es decir, los mensajes físicos que tienen números de secuencia de mensajes de 1, y desplazamientos de 0, determinan el orden en el que se devuelven los grupos elegibles.

La opción `MQGMO_LOGICAL_ORDER` afecta a las unidades de trabajo como se indica a continuación:

- Si el primer mensaje lógico o segmento de un grupo se recupera dentro de una unidad de trabajo, todos los demás mensajes lógicos y segmentos del grupo deben recuperarse dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola. Sin embargo, no es necesario recuperarlos dentro de la misma unidad de trabajo. Esto permite que un grupo de mensajes que consta de muchos mensajes físicos se divida entre dos o más unidades de trabajo consecutivas para el descriptor de contexto de cola.
- Si el primer mensaje lógico o segmento de un grupo no se recupera dentro de una unidad de trabajo y se utiliza el mismo descriptor de contexto de cola, ninguno de los otros mensajes lógicos y segmentos del grupo se puede recuperar dentro de una unidad de trabajo.

Si no se cumplen estas condiciones, la llamada `MQGET` falla con el código de razón `MQRC_INCONSISTENT_UOW`.

Cuando se especifica `MQGMO_LOGICAL_ORDER` , el `MQGMO` proporcionado en la llamada `MQGET` no debe ser menor que `MQGMO_VERSION_2`, y el `MQMD` no debe ser menor que `MQMD_VERSION_2`. Si no se cumple esta condición, la llamada falla con el código de razón `MQRC_WRONG_GMO_VERSION` o `MQRC_WRONG_MD_VERSION`, según corresponda.

Si no se especifica `MQGMO_LOGICAL_ORDER` para llamadas `MQGET` sucesivas para el descriptor de contexto de cola, los mensajes se devuelven sin tener en cuenta si pertenecen a grupos de mensajes o si son segmentos de mensajes lógicos. Esto significa que los mensajes o segmentos de un grupo o mensaje lógico determinado pueden devolverse desordenados, o mezclados con mensajes o segmentos de otros grupos o mensajes lógicos, o con mensajes que no están en grupos y no son segmentos. En esta situación, los mensajes concretos devueltos por llamadas `MQGET` sucesivas se controlan mediante las opciones `MQMO_*` especificadas en dichas llamadas (consulte el campo *`MatchOptions`* descrito en "MQGMO-Opciones de obtención de mensajes" en la página 367 para obtener detalles de estas opciones).

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico en el medio, después de que se haya producido una anomalía del sistema. Cuando se reinicia el sistema, la aplicación puede establecer los campos `GroupId`, `MsgSeqNumber`, `Offsety` `MatchOptions` en los valores adecuados y, a continuación, emitir la llamada `MQGET` con `MQGMO_SYNCPOINT`

o MQGMO_NO_SYNCPOINT establecido, pero sin especificar MQGMO_LOGICAL_ORDER. Si esta llamada es satisfactoria, el gestor de colas conserva la información de grupo y segmento, y las llamadas MQGET subsiguientes que utilizan ese descriptor de contexto de cola pueden especificar MQGMO_LOGICAL_ORDER como normal.

La información de grupo y segmento que el gestor de colas retiene para la llamada MQGET es independiente de la información de grupo y segmento que retiene para la llamada MQPUT. Además, el gestor de colas conserva información separada para:

- Llamadas de MQGET que eliminan mensajes de la cola.
- Llamadas de MQGET que examinan mensajes en la cola.

Para cualquier descriptor de contexto de cola determinado, la aplicación puede combinar llamadas MQGET que especifiquen MQGMO_LOGICAL_ORDER con llamadas MQGET que no lo hagan. Sin embargo, tenga en cuenta los siguientes puntos:

- Si omite MQGMO_LOGICAL_ORDER, cada llamada MQGET satisfactoria hace que el gestor de colas establezca la información de grupo y segmento guardada en los valores correspondientes al mensaje devuelto; esto sustituye a la información de grupo y segmento existente retenida por el gestor de colas para el descriptor de contexto de cola. Sólo se modifica la información adecuada para la acción de la llamada (examinar o eliminar).
- Si omite MQGMO_LOGICAL_ORDER, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; la llamada podría tener éxito con un código de terminación MQCC_WARNING . La Tabla 496 en la página 394 muestra los distintos casos que se pueden presentar. En estos casos, si el código de terminación no es MQCC_OK, el código de razón es uno de los siguientes (según corresponda):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_UOW

  Nota: El gestor de colas no comprueba la información de grupo y segmento al examinar una cola o al cerrar una cola que se ha abierto para examinar pero no para entrar; en estos casos, el código de finalización siempre es MQCC_OK (suponiendo que no haya otros errores).

Tabla 496. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información de grupo y segmento

La llamada actual es	La llamada anterior era **MQGET** con **MQGMO_LOGICAL_ORDER**	La llamada anterior era **MQGET** sin **MQGMO_LOGICAL_ORDER**
MQGET con MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET sin MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE con un grupo o mensaje lógico sin terminar	MQCC_WARNING	MQCC_OK

Las aplicaciones que desean recuperar mensajes y segmentos en orden lógico se recomiendan para especificar MQGMO_LOGICAL_ORDER, ya que esta es la opción más sencilla de utilizar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, es posible que las aplicaciones especializadas necesiten más control que el proporcionado por la opción MQGMO_LOGICAL_ORDER , y esto se puede lograr no especificando esa opción. A continuación, la aplicación debe asegurarse de que los campos MsgId, CorrelId, GroupId, MsgSeqNumbery Offset en MQMD, y las opciones MQMO_* en MatchOptions en MQGMO, se hayan establecido correctamente, antes de cada llamada MQGET .

Por ejemplo, una aplicación que desea reenviar mensajes físicos que recibe, sin tener en cuenta si estos mensajes están en grupos o segmentos de mensajes lógicos, no debe especificar

MQGMO_LOGICAL_ORDER. En una red compleja con múltiples rutas entre los gestores de colas de emisión y recepción, los mensajes físicos pueden llegar fuera de secuencia. Al no especificar MQGMO_LOGICAL_ORDER, ni el MQPMO_LOGICAL_ORDER correspondiente en la llamada MQPUT, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llegue, sin tener que esperar a que llegue el siguiente en orden lógico.

Puede especificar MQGMO_LOGICAL_ORDER con cualquiera de las otras opciones de MQGMO_* y con varias de las opciones de MQMO_* en las circunstancias adecuadas (consulte la sección anterior).

- z/OS En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice de MQIT_GROUP_ID. Para colas compartidas, el objeto CFSTRUCT con el que se correlaciona la cola debe estar en CFLEVEL (3) o superior.
- Esta opción está soportada para todas las colas locales para las plataformas siguientes:
  - AIX AIX
  - Linux Linux
  - IBM i IBM i
  - Solaris Solaris
  - Windows Windows

  y para IBM MQ MQI clients conectados a estos sistemas.

MQGMO_COMPLETE_MSG

Sólo la llamada MQGET puede devolver un mensaje lógico completo. Si el mensaje lógico está segmentado, el gestor de colas vuelve a ensamblar los segmentos y devuelve el mensaje lógico completo a la aplicación; el hecho de que el mensaje lógico se haya segmentado no es aparente para la aplicación que lo recupera.

Nota: Esta es la única opción que hace que el gestor de colas vuelva a ensamblar segmentos de mensajes. Si no se especifica, los segmentos se devuelven individualmente a la aplicación si están presentes en la cola (y satisfacen los otros criterios de selección especificados en la llamada MQGET). Las aplicaciones que no desean recibir segmentos individuales siempre deben especificar MQGMO_COMPLETE_MSG.

Para utilizar esta opción, la aplicación debe proporcionar un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo o especificar la opción MQGMO_ACCEPT_TRUNCATED_MSG .

Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y todavía no han llegado), la especificación de MQGMO_COMPLETE_MSG impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos de mensajes siguen contribuyendo al valor del atributo de cola **CurrentQDepth** ; esto significa que es posible que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero.

Para los mensajes persistentes, el gestor de colas puede volver a ensamblar los segmentos sólo dentro de una unidad de trabajo:

- Si la llamada MQGET está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de reensamblaje, el gestor de colas restablece en la cola los segmentos que se han eliminado durante el reensamblaje. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas restituye la unidad de trabajo.

- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede volver a ensamblarse. Si el mensaje no necesita reensamblarse, la llamada puede seguir siendo satisfactoria. Pero si el mensaje requiere reensamblaje, la llamada falla con el código de razón MQRC_UOW_NOT_AVAILABLE.

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar el reensamblaje.

Cada mensaje físico que es un segmento tiene su propio descriptor de mensaje. Para los segmentos que constituyen un único mensaje lógico, la mayoría de los campos del descriptor de mensaje son los mismos para todos los segmentos del mensaje lógico; normalmente son sólo los campos MsgId, Offsety MsgFlags los que difieren entre los segmentos del mensaje lógico. Sin embargo, si un segmento se coloca en una cola de mensajes no entregados en un gestor de colas intermedio, el manejador DLQ recupera el mensaje especificando la opción MQGMO_CONVERT , y esto puede hacer que se cambie el juego de caracteres o la codificación del segmento. Si el manejador DLQ envía correctamente el segmento en su camino, es posible que el segmento tenga un juego de caracteres o una codificación que difiera de los otros segmentos del mensaje lógico cuando el segmento llegue al gestor de colas de destino.

Un mensaje lógico que consta de segmentos en los que los campos CodedCharSetId y Encoding difieren no puede ser reensamblado por el gestor de colas en un único mensaje lógico. En su lugar, el gestor de colas vuelve a ensamblar y devuelve los primeros segmentos consecutivos al principio del mensaje lógico que tienen los mismos identificadores de juego de caracteres y codificaciones, y la llamada MQGET se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_INCONSISTENT_CCSIDS o MQRC_INCONSISTENT_ENCODINGS, según corresponda. Esto sucede independientemente de si se especifica MQGMO_CONVERT . Para recuperar los segmentos restantes, la aplicación debe volver a emitir la llamada MQGET sin la opción MQGMO_COMPLETE_MSG , recuperando los segmentos uno por uno. MQGMO_LOGICAL_ORDER se puede utilizar para recuperar los segmentos restantes en orden.

Una aplicación que coloca segmentos también puede establecer otros campos en el descriptor de mensaje en valores que difieren entre segmentos. Sin embargo, no hay ninguna ventaja al hacerlo si la aplicación receptora utiliza MQGMO_COMPLETE_MSG para recuperar el mensaje lógico. Cuando el gestor de colas vuelve a ensamblar un mensaje lógico, devuelve en el descriptor de mensaje los valores del descriptor de mensaje para el primer segmento; la única excepción es el campo MsgFlags , que el gestor de colas establece para indicar que el mensaje reensamblado es el único segmento.

Si se especifica MQGMO_COMPLETE_MSG para un mensaje de informe, el gestor de colas realiza un proceso especial. El gestor de colas comprueba la cola para ver si todos los mensajes de informe de ese tipo de informe relacionados con los distintos segmentos del mensaje lógico están presentes en la cola. Si lo son, se pueden recuperar como un solo mensaje especificando MQGMO_COMPLETE_MSG. Para que esto sea posible, los mensajes de informe deben ser generados por un gestor de colas o un MCA que dé soporte a la segmentación, o la aplicación de origen debe solicitar al menos 100 bytes de datos de mensaje (es decir, se deben especificar las opciones MQRO_*_WITH_DATA o MQRO_*_WITH_FULL_DATA adecuadas). Si hay menos de la cantidad completa de datos de aplicación para un segmento, los bytes que faltan se sustituyen por nulos en el mensaje de informe devuelto.

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, el cursor para examinar debe estar situado en un mensaje cuyo campo *Offset* en MQMD tenga un valor de 0. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

MQGMO_COMPLETE_MSG implica MQGMO_ALL_SEGMENTS_AVAILABLE, que por lo tanto no es necesario especificar.

MQGMO_COMPLETE_MSG se puede especificar con cualquiera de las otras opciones MQGMO_* aparte de MQGMO_SYNCPOINT_IF_PERSISTENT, y con cualquiera de las opciones MQMO_* aparte de MQMO_MATCH_OFFSET.

- z/OS En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o superior.
- En las plataformas siguientes:
  - AIX AIX
  - IBM i IBM i
  - Linux Linux
  - Solaris Solaris
  - Windows Windows

  y para IBM MQ MQI clients conectado a estos sistemas, esta opción está soportada para todas las colas locales.

MQGMO_ALL_MSGS_AVAILABLE

Los mensajes de un grupo pasan a estar disponibles para su recuperación sólo cuando todos los mensajes del grupo están disponibles. Si la cola contiene grupos de mensajes con algunos de los mensajes que faltan (quizás porque se han retrasado en la red y todavía no han llegado), especificar MQGMO_ALL_MSGS_AVAILABLE impide la recuperación de mensajes que pertenecen a grupos incompletos. Sin embargo, estos mensajes siguen contribuyendo al valor del atributo de cola **CurrentQDepth** ; esto significa que es posible que no haya grupos de mensajes recuperables, aunque CurrentQDepth sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón MQRC_NO_MSG_AVAILABLE después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de MQGMO_ALL_MSGS_AVAILABLE depende de si también se especifica MQGMO_LOGICAL_ORDER :

- Si se especifican ambas opciones, MQGMO_ALL_MSGS_AVAILABLE sólo tiene efecto cuando no hay ningún grupo o mensaje lógico actual. Si hay un grupo actual o un mensaje lógico, se ignora MQGMO_ALL_MSGS_AVAILABLE . Esto significa que MQGMO_ALL_MSGS_AVAILABLE puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica MQGMO_ALL_MSGS_AVAILABLE sin MQGMO_LOGICAL_ORDER, MQGMO_ALL_MSGS_AVAILABLE siempre tiene un efecto. Esto significa que la opción debe desactivarse después de que el primer mensaje del grupo se haya eliminado de la cola, para poder eliminar los mensajes restantes del grupo.

La finalización satisfactoria de una llamada MQGET especificando MQGMO_ALL_MSGS_AVAILABLE significa que en el momento en que se emitió la llamada MQGET , todos los mensajes del grupo estaban en la cola. Sin embargo, tenga en cuenta que otras aplicaciones todavía pueden eliminar mensajes del grupo (el grupo no está bloqueado para la aplicación que recupera el primer mensaje del grupo).

Si omite esta opción, los mensajes que pertenecen a grupos se pueden recuperar incluso cuando el grupo está incompleto.

MQGMO_ALL_MSGS_AVAILABLE implica MQGMO_ALL_SEGMENTS_AVAILABLE, que por lo tanto no es necesario especificar.

MQGMO_ALL_MSGS_AVAILABLE se puede especificar con cualquiera de las otras opciones de MQGMO_* y con cualquiera de las opciones de MQMO_* .

- z/OS En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o superior.
- En las plataformas siguientes:

– AIX AIX

– IBM i IBM i

– Linux Linux

– Solaris Solaris

– Windows Windows

y para IBM MQ MQI clients conectado a estos sistemas, esta opción está soportada para todas las colas locales.

MQGMO_ALL_SEGMENTS_AVAILABLE

Los segmentos de un mensaje lógico pasan a estar disponibles para su recuperación sólo cuando todos los segmentos del mensaje lógico están disponibles. Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y todavía no han llegado), la especificación de `MQGMO_ALL_SEGMENTS_AVAILABLE` impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos siguen contribuyendo al valor del atributo de cola **`CurrentQDepth`** ; esto significa que es posible que no haya mensajes lógicos recuperables, aunque `CurrentQDepth` sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón `MQRC_NO_MSG_AVAILABLE` después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de `MQGMO_ALL_SEGMENTS_AVAILABLE` depende de si también se especifica `MQGMO_LOGICAL_ORDER` :

- Si se especifican ambas opciones, `MQGMO_ALL_SEGMENTS_AVAILABLE` sólo tiene efecto cuando no hay ningún mensaje lógico actual. Si hay un mensaje lógico actual, se ignora `MQGMO_ALL_SEGMENTS_AVAILABLE` . Esto significa que `MQGMO_ALL_SEGMENTS_AVAILABLE` puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica `MQGMO_ALL_SEGMENTS_AVAILABLE` sin `MQGMO_LOGICAL_ORDER`, `MQGMO_ALL_SEGMENTS_AVAILABLE` siempre tiene un efecto. Esto significa que la opción debe desactivarse después de que el primer segmento del mensaje lógico se haya eliminado de la cola, para poder eliminar los segmentos restantes del mensaje lógico.

Si no se especifica esta opción, los segmentos de mensaje se pueden recuperar incluso cuando el mensaje lógico está incompleto.

Aunque tanto `MQGMO_COMPLETE_MSG` como `MQGMO_ALL_SEGMENTS_AVAILABLE` requieren que todos los segmentos estén disponibles antes de que se pueda recuperar cualquiera de ellos, el primero devuelve el mensaje completo, mientras que el segundo permite que los segmentos se recuperen uno por uno.

Si se especifica `MQGMO_ALL_SEGMENTS_AVAILABLE` para un mensaje de informe, el gestor de colas comprueba la cola para ver si hay al menos un mensaje de informe para cada uno de los segmentos que componen el mensaje lógico completo. Si existe, se cumple la condición `MQGMO_ALL_SEGMENTS_AVAILABLE` . Sin embargo, el gestor de colas no comprueba el *tipo* de los mensajes de informe presentes, por lo que puede haber una combinación de tipos de informe en los mensajes de informe relacionados con los segmentos del mensaje lógico. Como resultado, el éxito de `MQGMO_ALL_SEGMENTS_AVAILABLE` no implica que `MQGMO_COMPLETE_MSG` tenga éxito. Si hay una combinación de tipos de informe presentes para los segmentos de un mensaje lógico determinado, estos mensajes de informe se deben recuperar uno por uno.

Puede especificar `MQGMO_ALL_SEGMENTS_AVAILABLE` con cualquiera de las otras opciones de `MQGMO_*` y con cualquiera de las opciones de `MQMO_*` .

- En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o superior.
- En las plataformas siguientes:

– AIX AIX

– IBM i IBM i

– Linux Linux

– Solaris Solaris

– Windows Windows

y para IBM MQ MQI clients conectado a estos sistemas, esta opción está soportada para todas las colas locales.

Opciones de propiedad

Las opciones siguientes hacen relación a las propiedades del mensaje:

MQGMO_PROPERTIES_AS_Q_DEF

Las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión), deben representarse tal como se define en el atributo de cola **PropertyControl** . Si se proporciona un `MsgHandle` , esta opción se ignora y las propiedades del mensaje están disponibles a través de `MsgHandle`, a menos que el valor del atributo de cola **PropertyControl** sea `MQPROP_FORCE_MQRFH2`.

Se trata de la acción predeterminada si no se especifican opciones de propiedad.

MQGMO_PROPERTIES_IN_HANDLE

Las propiedades del mensaje deben estar disponibles a través de `MsgHandle`. Si no se proporciona ningún manejador de mensajes, la llamada fallará con la razón `MQRC_HMSG_ERROR`.

Nota: Si el mensaje lo lee más adelante una aplicación que no crea un manejador de mensajes, el gestor de colas coloca las propiedades de mensaje en una estructura `MQRFH2` . Es posible que encuentre que la presencia de una cabecera `MQRFH2` inesperada interrumpe el comportamiento de una aplicación existente.

MQGMO_NO_PROPERTIES

No se recuperarán las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión). Si se proporciona un `MsgHandle` , se ignorará.

MQGMO_PROPERTIES_FORCE_MQRFH2

Las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión), deben representarse utilizando cabeceras `MQRFH2` . Esto proporciona compatibilidad con versiones anteriores para aplicaciones que esperan recuperar propiedades pero que no pueden cambiarse para utilizar manejadores de mensajes. Si se proporciona un `MsgHandle` , se ignora.

MQGMO_PROPERTIES_COMPATIBILITY

Si el mensaje contiene una propiedad con el prefijo **"mcd."**, **"jms."**, **"usr."**o **"mqext."**, todas las propiedades del mensaje se entregan a la aplicación en una cabecera `MQRFH2` . De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación.

Opción predeterminada

Si no se requiere ninguna de las opciones descritas, se puede utilizar la siguiente opción:

MQGMO_NONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. `MQGMO_NONE` ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial del campo `Options` es `MQGMO_NO_WAIT` más `MQGMO_PROPERTIES_AS_Q_DEF`.

WaitInterval (MQLONG)

Es el tiempo aproximado, expresado en milisegundos, que la llamada MQGET espera a que llegue un mensaje adecuado (es decir, un mensaje que cumpla los criterios de selección especificados en el parámetro **MsgDesc** de la llamada MQGET.

Importante: No hay espera, ni retardo, si un mensaje adecuado está disponible inmediatamente.

Consulte el campo *MsgId* descrito en "MQMD - Descriptor de mensaje" en la página 421 para obtener más detalles). Si no ha llegado ningún mensaje adecuado después de que haya transcurrido este tiempo, la llamada se completa con MQCC_FAILED y el código de razón MQRC_NO_MSG_AVAILABLE.

En z/OS, el periodo de tiempo que la llamada MQGET realmente espera se ve afectado por las consideraciones de carga del sistema y planificación de trabajo, y puede variar entre el valor especificado para *WaitInterval* y aproximadamente 100 milisegundos mayor que *WaitInterval*.

WaitInterval se utiliza junto con la opción MQGMO_WAIT o MQGMO_SET_SIGNAL. Se ignora si no se especifica ninguno de ellos. Si se especifica uno de estos valores, *WaitInterval* debe ser mayor o igual que cero, o el siguiente valor especial:

MQWI_UNLIMITED
Intervalo de espera ilimitado.

El valor inicial de este campo es `0`.

Signal1 (MQLONG)

Es un campo de entrada que sólo se utiliza junto con la opción MQGMO_SET_SIGNAL; identifica una señal que se debe entregar cuando hay un mensaje disponible.

Nota: El tipo de datos y el uso de este campo están determinados por el entorno; por este motivo, las aplicaciones que desee portar entre distintos entornos no deben utilizar señales.

- En z/OS, este campo debe contener la dirección de un bloque de control de sucesos (ECB). La aplicación debe borrar el ECB antes de emitir la llamada MQGET. El almacenamiento que contiene el ECB no debe liberarse hasta que se cierre la cola. El gestor de colas publica el BCE con uno de los códigos de terminación de señal descritos. Estos códigos de terminación se establecen en los bits 2 a 31 del ECB, el área definida en la macro de correlación de z/OS IHAECB como para un código de terminación de usuario.
- En todos los demás entornos, se trata de un campo reservado; su valor no es significativo.

Los códigos de terminación de señal son:

MQEC_MSG_LLEGADO
Ha llegado un mensaje adecuado a la cola. Este mensaje no se ha reservado para el llamante; se debe emitir una segunda solicitud MQGET, pero otra aplicación podría recuperar el mensaje antes de realizar la segunda solicitud.

MQEC_WAIT_INTERVAL_CADUCADO
El *WaitInterval* especificado ha caducado sin que llegue un mensaje adecuado.

MQEC_WAIT_CANCELADO
La espera se ha cancelado por una razón indeterminada (como la terminación del gestor de colas o la inhabilitación de la cola). Vuelva a emitir la solicitud si desea realizar un diagnóstico adicional.

MQEC_Q_MGR_QUIESCING
La espera se ha cancelado porque el gestor de colas ha entrado en el estado de desactivación temporal (se ha especificado MQGMO_FAIL_IF_QUIESCING en la llamada MQGET).

MQEC_CONNECTION_QUIESCING
La espera se ha cancelado porque la conexión ha entrado en el estado de desactivación temporal (se ha especificado MQGMO_FAIL_IF_QUIESCING en la llamada MQGET).

El valor inicial de este campo lo determina el entorno:

- En z/OS, el valor inicial es el puntero nulo.
- En todos los demás entornos, el valor inicial es `0`.

Signal2 (MQLONG)

Se trata de un campo de entrada que sólo se utiliza junto con la opción MQGMO_SET_SIGNAL. Es un campo reservado; su valor no es significativo.

El valor inicial de este campo es `0`.

ResolvedQName (MQCHAR48)

Es un campo de salida que el gestor de colas establece en el nombre local de la cola de la que se ha recuperado el mensaje, tal como se ha definido en el gestor de colas local. Esto es diferente del nombre utilizado para abrir la cola si:

- Se ha abierto una cola alias (en cuyo caso, se devuelve el nombre de la cola local a la que se ha resuelto el alias), o
- Se ha abierto una cola modelo (en cuyo caso, se devuelve el nombre de la cola local dinámica).

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MatchOptions (MQLONG)

Estas opciones permiten a la aplicación elegir qué campos del parámetro **MsgDesc** utilizar para seleccionar el mensaje devuelto por la llamada MQGET. La aplicación establece las opciones necesarias en este campo y, a continuación, establece los campos correspondientes en el parámetro **MsgDesc** en los valores necesarios para esos campos. Sólo los mensajes que tienen estos valores en MQMD para el mensaje son candidatos para la recuperación utilizando ese parámetro **MsgDesc** en la llamada MQGET. Los campos para los que no se ha especificado la opción de coincidencia correspondiente se ignoran al seleccionar el mensaje que se devolverá. Si no especifica ningún criterio de selección en la llamada MQGET (es decir, *cualquier* mensaje es aceptable), establezca *`MatchOptions`* en MQMO_NONE.

- En z/OS, los criterios de selección que se pueden utilizar pueden estar restringidos por el tipo de índice utilizado para la cola. Consulte el atributo de cola **IndexType** para obtener más detalles.

Si especifica MQGMO_LOGICAL_ORDER, sólo determinados mensajes pueden ser devueltos por la siguiente llamada MQGET:

- Si no hay ningún grupo actual o mensaje lógico, sólo se podrán devolver los mensajes que tengan *`MsgSeqNumber`* igual a `1` y *`Offset`* igual a `0` . En esta situación, puede utilizar una o más de las siguientes opciones de coincidencia para seleccionar cuál de los mensajes elegibles se devuelve:
  - MQMO_MATCH_MSG_ID
  - MQMO_MATCH_CORREL_ID
  - MQMO_MATCH_GROUP_ID
- Si hay un grupo o mensaje lógico actual, sólo el siguiente mensaje del grupo o segmento siguiente del mensaje lógico es elegible para su devolución, y esto no se puede modificar especificando las opciones MQMO_ *.

En los dos casos anteriores, puede especificar opciones de coincidencia que no se aplican, pero el valor del campo relevante en el parámetro **MsgDesc** debe coincidir con el valor del campo correspondiente en el mensaje que se va a devolver; la llamada falla con el código de razón MQRC_MATCH_OPTIONS_ERROR es que esta condición no se cumple.

`MatchOptions` se ignora si especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR.

La obtención de mensajes basados en la propiedad de mensaje no se realiza utilizando opciones de coincidencia; para obtener más información, consulte “SelectionString (MQCHARV)” en la página 496.

Puede especificar una o más de las siguientes opciones de coincidencia:

MQMO_MATCH_MSG_ID

El mensaje que se va a recuperar debe tener un identificador de mensaje que coincida con el valor del campo *MsgId* en el parámetro **MsgDesc** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, el campo *MsgId* en el parámetro **MsgDesc** se ignora y cualquier identificador de mensaje coincidirá.

Nota: El identificador de mensaje MQMI_NONE es un valor especial que coincide con cualquier identificador de mensaje del MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_MSG_ID con MQMI_NONE es lo mismo que no especificar MQMO_MATCH_MSG_ID.

MQMO_MATCH_CORREL_ID

El mensaje que se va a recuperar debe tener un identificador de correlación que coincida con el valor del campo *CorrelId* en el parámetro **MsgDesc** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de mensaje).

Si omite esta opción, el campo *CorrelId* del parámetro **MsgDesc** se ignora y cualquier identificador de correlación coincidirá.

Nota: El identificador de correlación MQCI_NONE es un valor especial que coincide con *cualquier* identificador de correlación en el MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_CORREL_ID con MQCI_NONE es lo mismo que no especificar MQMO_MATCH_CORREL_ID.

MQMO_MATCH_GROUP_ID

El mensaje que se va a recuperar debe tener un identificador de grupo que coincida con el valor del campo *GroupId* en el parámetro **MsgDesc** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, el campo *GroupId* del parámetro **MsgDesc** se ignora y cualquier identificador de grupo coincidirá.

Nota: El identificador de grupo MQGI_NONE es un valor especial que coincide con *cualquier* identificador de grupo en el MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_GROUP_ID con MQGI_NONE es lo mismo que no especificar MQMO_MATCH_GROUP_ID.

MQMO_MATCH_MSG_SEQ_NUMBER

El mensaje que se va a recuperar debe tener un número de secuencia de mensaje que coincida con el valor del campo *MsgSeqNumber* en el parámetro **MsgDesc** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de grupo).

Si omite esta opción, el campo *MsgSeqNumber* del parámetro **MsgDesc** se ignora y cualquier número de secuencia de mensaje coincidirá.

MQMO_MATCH_OFFSET

El mensaje que se va a recuperar debe tener un desplazamiento que coincida con el valor del campo *Offset* en el parámetro **MsgDesc** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que se pueda aplicar (por ejemplo, el número de secuencia de mensaje).

Si omite esta opción no se especifica, el campo *Offset* del parámetro **MsgDesc** se ignora y cualquier desplazamiento coincidirá.

- Esta opción no está soportada en z/OS.

MQMO_MATCH_MSG_TOKEN

El mensaje que se va a recuperar debe tener una señal de mensaje que coincida con el valor del campo *MsgToken* en la estructura MQGMO especificada en la llamada MQGET.

Puede especificar esta opción para todas las colas locales. Si lo especifica para una cola que tiene un *IndexType* de MQIT_MSG_TOKEN (una cola gestionada por WLM), no puede especificar ninguna otra opción de coincidencia con MQMO_MATCH_MSG_TOKEN.

No puede especificar MQMO_MATCH_MSG_TOKEN con MQGMO_WAIT o MQGMO_SET_SIGNAL. Si la aplicación desea esperar a que llegue un mensaje a una cola que tenga un *IndexType* de MQIT_MSG_TOKEN, especifique MQMO_NONE.

Si omite esta opción, el campo *MsgToken* en MQGMO se ignora y cualquier señal de mensaje coincidirá.

Si no especifica ninguna de las opciones descritas, puede utilizar la opción siguiente:

MQMO_NONE
No utilice coincidencias al seleccionar el mensaje que se va a devolver; todos los mensajes de la cola son elegibles para la recuperación (pero están sujetos al control de las opciones MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE y MQGMO_COMPLETE_MSG).

MQMO_NONE ayuda a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción MQMO_ *, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial de este campo es MQMO_MATCH_MSG_ID con MQMO_MATCH_CORREL_ID. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Nota: El valor inicial del campo *MatchOptions* se define para la compatibilidad con gestores de colas anteriores de MQSeries . Sin embargo, al leer una serie de mensajes de una cola sin utilizar criterios de selección, este valor inicial requiere que la aplicación restablezca los campos *MsgId* y *CorrelId* en MQMI_NONE y MQCI_NONE antes de cada llamada MQGET. Evite la necesidad de restablecer *MsgId* y *CorrelId* estableciendo *Version* en MQGMO_VERSION_2y *MatchOptions* en MQMO_NONE.

Conceptos relacionados
Selectores de mensajes en JMS

GroupStatus (MQCHAR)

Este distintivo indica si el mensaje recuperado está en un grupo.

Tiene uno de los siguientes valores:

MQGS_NOT_IN_GROUP
El mensaje no está en un grupo.

MQGS_MSG_IN_GROUP
El mensaje está en un grupo, pero no es el último del grupo.

MQGS_LAST_MSG_IN_GROUP
El mensaje es el último del grupo.

También es el valor devuelto si el grupo consta de un solo mensaje.

Se trata de un campo de salida. El valor inicial de este campo es MQGS_NOT_IN_GROUP. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

SegmentStatus (MQCHAR)

Es un distintivo que indica si el mensaje recuperado es un segmento de un mensaje lógico. Tiene uno de los siguientes valores:

MQSS_NO_A_SEGMENTO
El mensaje no es un segmento.

SEGMENTO_MQSS
El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

MQSS_LAST_SEGMENT
El mensaje es el último segmento del mensaje lógico.

También es el valor devuelto si el mensaje lógico consta de un solo segmento.

En z/OS, el gestor de colas siempre establece este campo en MQSS_NOT_A_SEGMENT.

Se trata de un campo de salida. El valor inicial de este campo es MQSS_NOT_A_SEGMENT. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Segmentación (MQCHAR)

Es un distintivo que indica si se permite una segmentación adicional para el mensaje recuperado. Tiene uno de los siguientes valores:

MQSEG_INHIBIDO
Segmentación no permitida.

MQSEG_PERMITIDO
Segmentación permitida.

En z/OS, el gestor de colas siempre establece este campo en MQSEG_INHIBIDO.

Se trata de un campo de salida. El valor inicial de este campo es MQSEG_INHIBIDO. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

MsgToken (MQBYTE16)

Campo MsgToken -Estructura MQGMO. El gestor de colas utiliza este campo para identificar de forma exclusiva un mensaje.

Es una serie de bytes generada por el gestor de colas para identificar un mensaje de forma exclusiva en una cola. La señal de mensaje se genera cuando el mensaje se coloca por primera vez en el gestor de colas y permanece con el mensaje hasta que el mensaje se elimina permanentemente del gestor de colas, a menos que se reinicie el gestor de colas.

Cuando se elimina el mensaje de la cola, el *MsgToken* que ha identificado esa instancia del mensaje ya no es válido y nunca se reutiliza. Si se reinicia el gestor de colas, es posible que el *MsgToken* que ha identificado un mensaje en la cola antes del reinicio no sea válido después del reinicio. Sin embargo, el *MsgToken* nunca se reutiliza para identificar una instancia de mensaje diferente. El *MsgToken* lo genera el gestor de colas y no es visible para ninguna aplicación externa.

Cuando una llamada a MQGET devuelve un mensaje donde se proporciona un MQGMO de la versión 3 o superior, el gestor de colas devuelve el *MsgToken* que identifica el mensaje en la cola en el MQGMO. Hay una excepción a esto: cuando el mensaje se elimina de la cola fuera del punto de sincronización, es posible que el gestor de colas no devuelva un *MsgToken* porque no es útil identificar el mensaje devuelto en una llamada MQGET posterior. Las aplicaciones sólo deben utilizar *MsgToken* para hacer referencia al mensaje en las llamadas MQGET posteriores.

Si se proporciona un *MsgToken* y se especifica *MatchOption* MQMO_MATCH_MSG_TOKEN y no se especifica MQGMO_MSG_UNDER_CURSOR ni MQGMO_BROWSE_MSG_UNDER_CURSOR, sólo se puede devolver el mensaje identificado por ese *MsgToken* . La opción es válida en todas las colas locales independientemente de INDXTYPE, y en z/OS debe utilizar INDXTYPE (MSGTOKEN) sólo en colas del Gestor de carga de trabajo (WLM).

Los demás *MatchOptions* especificados se comprueban y, si no coinciden, se devuelve MQRC_NO_MSG_AVAILABLE. Si MQGMO_BROWSE_NEXT está codificado con MQMO_MATCH_MSG_TOKEN, el mensaje identificado por *MsgToken* sólo se devuelve si está más allá del cursor para examinar para el descriptor de contexto de llamada.

Si se especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, se ignora MQMO_MATCH_MSG_TOKEN.

MQMO_MATCH_MSG_TOKEN no es válido con las siguientes opciones de obtención de mensaje:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

Para una llamada MQGET que especifique MQMO_MATCH_MSG_TOKEN, se debe proporcionar un MQGMO de la versión 3 o posterior a la llamada, de lo contrario se devuelve MQRC_INJU_GMO_VERSION.

Si *MsgToken* no es válido en este momento, se devuelve MQCC_FAILED con MQRC_NO_MSG_AVAILABLE, a menos que haya otro error.

ReturnedLength (MQLONG)

Es un campo de salida que el gestor de colas establece en la longitud, en bytes, de los datos de mensaje devueltos por la llamada MQGET en el parámetro **Buffer** . Si el gestor de colas no da soporte a esta prestación, *ReturnedLength* se establece en el valor MQRL_UNDEFINED.

Cuando los mensajes se convierten entre codificaciones o juegos de caracteres, los datos del mensaje a veces pueden cambiar de tamaño. A la devolución de la llamada MQGET:

- Si *ReturnedLength* no es MQRL_UNDEFINED, el número de bytes de datos de mensaje devueltos lo proporciona *ReturnedLength*.
- Si *ReturnedLength* tiene el valor MQRL_UNDEFINED, el número de bytes de datos de mensaje devueltos normalmente lo proporciona el menor de *BufferLength* y *DataLength*, pero puede ser *menor que* si la llamada MQGET se completa con el código de razón MQRC_TRUNCATED_MSG_ACCEPTED. Si esto sucede, los bytes insignificantes del parámetro **Buffer** se establecen en nulos.

Se define el siguiente valor especial:

MQRL_UNDEFINED
No se ha definido la longitud de los datos devueltos.

En z/OS, el valor devuelto para el campo *ReturnedLength* es siempre MQRL_UNDEFINED.

El valor inicial de este campo es MQRL_UNDEFINED. Este campo se ignora si *Version* es menor que MQGMO_VERSION_3.

Reserved2 (MQLONG)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que **MQGMO_VERSION_4**.

MsgHandle (MQHMSG)

Si se especifica la opción MQGMO_PROPERTIES_AS_Q_DEF y el atributo de cola `PropertyControl` no está establecido en MQPROP_FORCE_MQRFH2 , este es el descriptor de contexto de un mensaje que se rellenará con las propiedades del mensaje que se está recuperando de la cola. El descriptor de contexto se crea mediante una llamada MQCRTMH. Las propiedades que ya estén asociadas con el descriptor de contexto se borrarán antes de recuperar un mensaje.

También se puede especificar el valor siguiente:
MQHM_NONE
No se ha proporcionado ningún manejador de mensajes.

No es necesario ningún descriptor de mensaje en la llamada MQGET si se proporciona un descriptor de mensaje válido y se utiliza en la salida para contener las propiedades del mensaje, el descriptor de mensaje asociado con el descriptor de mensaje se utiliza para los campos de entrada.

Si se especifica un descriptor de mensaje en la llamada MQGET, siempre tiene prioridad sobre el descriptor de mensaje asociado a un descriptor de mensaje.

Si se especifica MQGMO_PROPERTIES_FORCE_MQRFH2 , o se especifica MQGMO_PROPERTIES_AS_Q_DEF y el atributo de cola `PropertyControl` es MQPROP_FORCE_MQRFH2 , la llamada falla con el código de razón MQRC_MD_ERROR cuando no se especifica ningún parámetro de descriptor de mensaje.

Al volver de la llamada MQGET, las propiedades y el descriptor de mensaje asociados con este descriptor de mensaje se actualizan para reflejar el estado del mensaje recuperado (así como el descriptor de mensaje si se ha proporcionado uno en la llamada MQGET). A continuación, se pueden consultar las propiedades del mensaje utilizando la llamada MQINQMP.

Excepto para las extensiones de descriptor de mensaje, cuando están presentes, una propiedad que se puede consultar con la llamada MQINQMP no está contenida en los datos del mensaje; si el mensaje de la cola contenía propiedades en los datos del mensaje, estos se eliminan de los datos del mensaje antes de que se devuelvan los datos a la aplicación.

Si no se proporciona ningún descriptor de mensaje o la versión es inferior a MQGMO_VERSION_4 , debe proporcionar un descriptor de mensaje válido en la llamada MQGET. Las propiedades de mensaje (excepto las contenidas en el descriptor de mensaje) se devuelven en el asunto de datos de mensaje al valor de las opciones de propiedad en la estructura MQGMO y el atributo de cola `PropertyControl` .

Se trata de un campo de entrada siempre. El valor inicial de este campo es MQHM_NONE. Este campo se ignora si `Version` es menor que MQGMO_VERSION_4.

MQIIH – Cabecera información de IMS

La estructura MQIIH describe la información de cabecera de un mensaje enviado a IMS a través del puente IMS .Para cualquier plataforma soportada por IBM MQ , puede crear y transmitir un mensaje que incluya la estructura MQIIH, pero sólo un gestor de colas IBM MQ for z/OS puede utilizar el puente IMS . Por lo tanto, para que el mensaje llegue a IMS desde un gestor de colas noz/OS , la red de gestores de colas debe incluir al menos un gestor de colas z/OS a través del cual se pueda direccionar el mensaje.

Disponibilidad

Todos los sistemas IBM MQ y clientes IBM MQ .

Nombre de formato

MQFMT_IMS

Juego de caracteres y codificación

Se aplican condiciones especiales al juego de caracteres y la codificación utilizados para la estructura MQIIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente IMS deben proporcionar una estructura MQIIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQIIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQIIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola puente IMS convierte la MQIIH.
- Los datos del mensaje de aplicación que siguen a la estructura MQIIH deben estar en el mismo juego de caracteres y codificación que la estructura MQIIH. No utilice los campos *`CodedCharSetId`* y *`Encoding`* en la estructura MQIIH para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

  Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 497. Campos en MQIIH para MQIIH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQIIH_STRUC_ID	`'IIH¬'`

Tabla 497. Campos en MQIIH para MQIIH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
Versión (número de versión de estructura)	MQIIH_VERSION_1	1
StrucLength (longitud de la estructura MQIIH)	MQIIH_LENGTH_1	84
Codificación (reservado-consulte “Juego de caracteres y codificación” en la página 406)	Ninguna	0
CodedCharSetId (reservado-consulte “Juego de caracteres y codificación” en la página 406)	Ninguna	0
Format (nombre de formato de datos deMQ que sigue a MQIIH)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQIIH_NONE	0
LTermOverride (alteración temporal de terminal lógico)	Ninguna	Espacios en blanco
MFSMapName (nombre de correlación de servicios de formato de mensaje)	Ninguna	Espacios en blanco
FormatoReplyTo (nombre de formatoMQ del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
Autenticador (contraseña o passticket deRACF)	MQIAUT_NONE	Espacios en blanco
TranInstanceId (identificador de instancia de transacción)	MQITII_NONE	Nulos
TranState (estado de transacción)	MQITS_NO_EN_CONVERSACIÓN	'¬'
CommitMode (modalidad de confirmación)	MQICM_COMMIT_THEN_SEND	'0'
SecurityScope (ámbito de seguridad)	MQISS_CHECK	'C'
Reservado (reservado)	Ninguna	'¬'

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQIIH_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Es el identificador de estructura. El valor debe ser:

MQIIH_STRUC_ID
Identificador de la estructura de cabecera de información de IMS .

Para el lenguaje de programación C, también se define la constante MQIIH_STRUC_ID_ARRAY; tiene el mismo valor que MQIIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQIIH_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQIIH_VERSION_1
Número de versión para la estructura de cabecera de información de IMS .

La constante siguiente especifica el número de versión de la versión actual:

MQIIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de IMS .

El valor inicial de este campo es MQIIH_VERSION_1.

StrucLength (MQLONG)

Es la longitud de la estructura MQIIH. El valor debe ser:

MQIIH_LENGTH_1
Longitud de la estructura de cabecera de información de IMS .

El valor inicial de este campo es MQIIH_LENGTH_1.

Encoding (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

La codificación para las estructuras soportadas que siguen a una estructura MQIIH es la misma que la de la propia estructura MQIIH y se toma de cualquier cabecera MQ anterior.

CodedCharSetId (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

El ID de juego de caracteres para las estructuras soportadas que siguen a una estructura MQIIH es el mismo que el de la propia estructura MQIIH y se toma de cualquier cabecera MQ anterior.

Format (MQCHAR8)

Especifica el nombre de formato MQ de los datos que siguen a la estructura MQIIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor de distintivos debe ser:

MQIIH_NONE
Sin distintivos.

MQIIH_PASS_EXPIRATION
El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud
- El tiempo de caducidad restante del mensaje de solicitud sin realizar ningún ajuste para el tiempo de proceso del puente

Si no se establece este valor, la hora de caducidad se establece en *unlimited*.

MQIIH_REPLY_FORMAT_NONE
Establece MQIIH.Format de la respuesta a MQFMT_NONE.

MQIIH_IGNORE_PURG
Establece el indicador TMAMIPRG en el prefijo OTMA, que solicita que OTMA ignore las llamadas PURG en el TP PCB para las transacciones CM0 .

MQIIH_CM0_REQUEST_RESPONSE
Para transacciones de modalidad de confirmación 0 (CM0), este distintivo establece el indicador TMAMHRSP en el prefijo OTMA. Al establecer este indicador se solicita que OTMA/IMS genere un mensaje DFS2082 RESPONSE MODE TRANSACTION TERMINADO WITHOUT REPLY cuando el programa de aplicación IMS original no responde al IOPCB ni conmuta el mensaje a otra transacción.

El valor inicial de este campo es MQIIH_NONE.

LTermOverride (MQCHAR8)

La alteración temporal de terminal lógico, situada en el campo PCB de E/S. Es opcional; si no se especifica, se utiliza el nombre TPIPE. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona MQ_LTERM_OVERRIDE_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

MFSMapName (MQCHAR8)

El nombre de correlación de servicios de formato de mensaje, situado en el campo PCB de E/S. Es opcional. En la entrada representa el MID, en la salida representa el MOD. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona MQ_MFS_MAP_NAME_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

Formato ReplyTo(MQCHAR8)

Es el nombre de formato MQ del mensaje de respuesta que se envía en respuesta al mensaje actual. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Para convertir los datos en el mensaje de respuesta utilizando MQGMO_CONVERT, especifique MQIIH.replyToFormat= MQFMT_STRING o MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING. Para obtener una explicación del uso de estos campos, consulte “Format (MQCHAR8)” en la página 448.

Si se utiliza el valor predeterminado (MQIIH.replyToFormat= MQFMT_NONE) en el mensaje de solicitud y el mensaje de respuesta se recupera utilizando MQGMO_CONVERT, no se realiza ninguna conversión de datos.

Autenticador (MQCHAR8)

Esta es la contraseña de RACF o PassTicket. Es opcional; si se especifica, se utiliza con el ID de usuario en el contexto de seguridad MQMD para crear un UTOKEN que se envía a IMS para proporcionar un contexto de seguridad. Si no se especifica, el ID de usuario se utiliza sin verificación. Esto depende del valor de los conmutadores RACF , que pueden requerir la presencia de un autenticador.

Esto se ignora si el primer byte está en blanco o es nulo. Se puede utilizar el siguiente valor especial:

MQIAUT_NONE
Sin autenticación.

Para el lenguaje de programación C, la constante MQIAUT_NONE_ARRAY también está definida; tiene el mismo valor que MQIAUT_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es MQIAUT_NONE.

TranInstanceId (MQBYTE16)

Es el identificador de instancia de transacción. Este campo lo utilizan los mensajes de salida de IMS, por lo que se ignora en la primera entrada. Si establece *TranState* en MQITS_IN_CONVERSATION, esto se debe proporcionar en la siguiente entrada, y en todas las entradas posteriores, para permitir que IMS correlacione los mensajes con la conversación correcta. Puede utilizar el siguiente valor especial:

MQITII_NONE
No hay ningún identificador de instancia de transacción.

Para el lenguaje de programación C, también se define la constante MQITII_NONE_ARRAY; tiene el mismo valor que MQITII_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQITII_NONE.

TranState (MQCHAR)

Indica el estado de conversación de IMS . Esto se ignora en la primera entrada porque no existe ninguna conversación. En entradas posteriores, indica si una conversación está activa o no. En la salida se establece mediante IMS. El valor debe ser uno de los siguientes:

MQITS_EN_CONVERSACIÓN
En conversación.

MQITS_NO_EN_CONVERSACIÓN
No en conversación.

MQITS_ARCHITECTED
Devolver datos de estado de transacción en formato de arquitectura.

Este valor sólo se utiliza con el mandato IMS `/DISPLAY TRAN` . Devuelve los datos de estado de transacción en el formato de arquitectura IMS en lugar del formato de caracteres. z/OS Para obtener más información, consulte Escritura de programas de transacción de IMS a través de IBM MQ.

El valor inicial de este campo es MQITS_NOT_IN_CONVERSATION.

CommitMode (MQCHAR)

Esta es la modalidad de confirmación de IMS . Consulte la publicación *OTMA Reference* para obtener más información sobre las modalidades de confirmación de IMS . El valor debe ser uno de los siguientes:

MQICM_COMMIT_THEN_SEND
Confirme y, a continuación, envíe.

Esta modalidad implica una doble cola de salida, pero tiempos de ocupación de región más cortos. Las transacciones de vía de acceso rápida y conversacional no se pueden ejecutar con esta modalidad.

MQICM_SEND_THEN_COMMIT
Enviar y, a continuación, confirmar.

Cualquier transacción IMS iniciada como resultado de una modalidad de confirmación de MQICM_SEND_THEN_COMMIT se ejecuta en modalidad RESPONSE independientemente de cómo se haya definido la transacción en la definición del sistema IMS (parámetro MSGTYPE en la macro TRANSACT). Esto también se aplica a las transacciones iniciadas mediante un conmutador de transacción.

El valor inicial de este campo es MQICM_COMMIT_THEN_SEND.

SecurityScope (MQCHAR)

Esto indica el proceso de seguridad de IMS necesario. Los valores siguientes están definidos:

MQISS_CHECK
Comprobar ámbito de seguridad: se crea un ACEE en la región de control, pero no en la región dependiente.

MQISS_FULL
Ámbito de seguridad completo: un ACEE almacenado en memoria caché se crea en la región de control y un ACEE no almacenado en memoria caché se crea en la región dependiente. Si utiliza MQISS_FULL, asegúrese de que el ID de usuario para el que se crea el ACEE tenga acceso a los recursos utilizados en la región dependiente.

Si no se especifica MQISS_CHECK ni MQISS_FULL para este campo, se presupone MQISS_CHECK.

El valor inicial de este campo es MQISS_CHECK.

Reservado (MQCHAR)

Es un campo reservado; debe estar en blanco.

MQIMPO-Consultar opciones de propiedad de mensaje

La estructura MQIMPO permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQINQMP.

Disponibilidad

Todos los sistemas IBM MQ y clientes IBM MQ .

Juego de caracteres y codificación

Los datos de MQIMPO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 498. Campos en MQIPMO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQIM_ID_ESTRUCTURA	'IMPO'
Versión (número de versión de estructura)	MQIMPO_VERSION_1	1
Options (opciones que controlan la acción de MQINQMP)	MQIM_INQ_PRIMERO	
RequestedEncoding (codificación en la que se va a convertir la propiedad consultada)	MQENC_NATIVE	
RequestedCCSID (juego de caracteres de la propiedad consultada)	MQCCSI_APPL	
ReturnedEncoding (codificación del valor devuelto)	MQENC_NATIVE	
ReturnedCCSID	0	
Reserved1 (campo reservado)	carácter en blanco (campo de 4 bytes)	
ReturnedName (nombre de la propiedad consultada)	MQCHARV_PREDETERMINADO	
TypeString (representación de serie del tipo de datos de la propiedad)	Serie nula o espacios en blanco	

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQIMPO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Estructura de opciones de propiedad de mensaje de consulta-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQIM_ID_ESTRUCTURA
Identificador de la estructura de opciones de propiedad de mensaje de consulta.

Para el lenguaje de programación C, también se define la constante MQIMPO_STRUC_ID_ARRAY; tiene el mismo valor que MQIMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_STRUC_ID.

Versión (MQLONG)

Consultar estructura de opciones de propiedad de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQIMPO_VERSION_1
Número de versión para la estructura de opciones de propiedad de mensaje de consulta.

La constante siguiente especifica el número de versión de la versión actual:

MQIM_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de propiedad de mensaje de consulta.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_VERSION_1.

Opciones (MQLONG)

Estructura de opciones de propiedad de mensaje de consulta-Campo Opciones

Las opciones siguientes controlan la acción de MQINQMP. Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

Las combinaciones de opciones que no son válidas se anotan; todas las demás combinaciones son válidas.

Opciones de datos de valor: Las opciones siguientes están relacionadas con el proceso de los datos de valor cuando se recupera la propiedad del mensaje.

MQIM_CONVERT_VALUE
Esta opción solicita que el valor de la propiedad se convierta para que se ajuste a los valores *RequestedCCSID* y *RequestedEncoding* especificados antes de que la llamada MQINQMP devuelva el valor de la propiedad en el área *Value* .

- Si la conversión se realiza correctamente, los campos *ReturnedCCSID* y *ReturnedEncoding* se establecen en los mismos que *RequestedCCSID* y *RequestedEncoding* al volver de la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP de lo contrario se completa sin error, el valor de propiedad se devuelve sin convertir.

  Si la propiedad es una serie, los campos *ReturnedCCSID* y *ReturnedEncoding* se establecen en el juego de caracteres y la codificación de la serie no convertida.

  El código de terminación es MQCC_WARNING en este caso, con el código de razón MQRC_PROP_VALUE_NOT_CONVERTED. El cursor de propiedad se ha avanzado a la propiedad devuelta.

Si el valor de propiedad se expande durante la conversión y supera el tamaño del parámetro **Value** , el valor se devuelve sin convertir, con el código de terminación MQCC_FAILED; el código de razón se establece en MQRC_PROPERTY_VALUE_TOO_BIG.

El parámetro **DataLength** de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Esta opción también solicita que:

- Si el nombre de propiedad contiene un comodín, y
- El campo *ReturnedName* se inicializa con una dirección o desplazamiento para el nombre devuelto,

entonces el nombre devuelto se convierte para ajustarse a los valores *RequestedCCSID* y *RequestedEncoding* .

- Si la conversión es satisfactoria, el campo *VSCCSID* de *ReturnedName* y la codificación del nombre devuelto se establecen en el valor de entrada de *RequestedCCSID* y *RequestedEncoding*.
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin error ni aviso, el nombre devuelto no se convierte. El código de terminación es MQCC_WARNING en este caso, con el código de razón MQRC_PROP_NAME_NOT_CONVERT.

  El cursor de propiedad se ha avanzado a la propiedad devuelta. Se devuelve MQRC_PROP_VALUE_NOT_CONVERTED si no se convierten tanto el valor como el nombre.

Si el nombre devuelto se expande durante la conversión y supera el tamaño del campo *VSBufsize* de *RequestedName*, la serie devuelta se deja sin convertir, con el código de terminación MQCC_FAILED y el código de razón se establece en MQRC_PROPERTY_NAME_TOO_BIG.

El campo *VSLength* de la estructura MQCHARV devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

MQIM_CONVERT_TYPE

Esta opción solicita que el valor de la propiedad se convierta de su tipo de datos actual en el tipo de datos especificado en el parámetro **Type** de la llamada MQINQMP.

- Si la conversión es satisfactoria, el parámetro **Type** no se modifica al devolver la llamada MQINQMP.
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin errores, la llamada falla con la razón MQRC_PROP_CONV_NOT_SUPPORTED. El cursor de propiedad no se ha modificado.

Si la conversión del tipo de datos hace que el valor se expanda durante la conversión y el valor convertido supera el tamaño del parámetro **Value** , el valor se devuelve sin convertir, con el código de terminación MQCC_FAILED y el código de razón se establece en MQRC_PROPERTY_VALUE_TOO_BIG.

El parámetro **DataLength** de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento

intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Si el valor del parámetro **Type** de la llamada MQINQMP no es válido, la llamada falla con la razón MQRC_PROPERTY_TYPE_ERROR.

Si la conversión de tipo de datos solicitada no está soportada, la llamada falla con la razón MQRC_PROP_CONV_NOT_SUPPORTED. Se da soporte a las siguientes conversiones de tipos de datos:

Tabla 499. Conversiones de tipos de datos soportadas

Tipo de datos de propiedad	Tipos de datos de destino soportados
MQTYPE_BOOLEAN	MQTYPE_STRING, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_BYTE_STRING	MQTYPE_STRING
MQTYPE_INT8	MQTYPE_STRING, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT16	MQTYPE_STRING, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT32	MQTYPE_STRING, MQTYPE_INT64
MQTYPE_INT64	MQTYPE_STRING
MQTYPE_FLOAT32	MQTYPE_STRING, MQTYPE_FLOAT64
MQTYPE_FLOAT64	MQTYPE_STRING
MQTYPE_STRING	MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64
MQTYPE_NULL	Ninguna

Las normas generales que rigen las conversiones soportadas son las siguientes:

- Los valores de propiedad numéricos se pueden convertir de un tipo de datos a otro, siempre que no se pierdan datos durante la conversión.

  Por ejemplo, el valor de una propiedad con el tipo de datos MQTYPE_INT32 se puede convertir en un valor con el tipo de datos MQTYPE_INT64, pero no se puede convertir en un valor con el tipo de datos MQTYPE_INT16.
- Un valor de propiedad de cualquier tipo de datos se puede convertir a una serie.
- Un valor de propiedad de serie se puede convertir a cualquier otro tipo de datos, siempre que la serie tenga el formato correcto para la conversión. Si una aplicación intenta convertir un valor de propiedad de serie que no tiene el formato correcto, IBM MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR.
- Si una aplicación intenta una conversión que no está soportada, IBM MQ devuelve el código de razón MQRC_PROP_CONV_NOT_SUPPORTED.

Las reglas específicas para convertir un valor de propiedad de un tipo de datos a otro son las siguientes:

- Al convertir un valor de propiedad MQTYPE_BOOLEAN en una serie, el valor TRUE se convierte a la serie "TRUE" y el valor false se convierte a la serie "FALSE".
- Al convertir un valor de propiedad MQTYPE_BOOLEAN en un tipo de datos numérico, el valor TRUE se convierte en uno y el valor FALSE se convierte en cero.
- Al convertir un valor de propiedad de serie a un valor MQTYPE_BOOLEAN, la serie "TRUE", o "1", se convierte a TRUE, y la serie "FALSE", o "0", se convierte a FALSE.

  Tenga en cuenta que los términos "TRUE" y "FALSE" no distinguen entre mayúsculas y minúsculas.

Cualquier otra serie no se puede convertir; IBM MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR.

- Al convertir un valor de propiedad de serie a un valor con el tipo de datos MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 o MQTYPE_INT64, la serie debe tener el formato siguiente:

```
[blanks][sign]digits
```

El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

Después de la secuencia de caracteres de dígitos, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por sentado que la serie representa un entero decimal.

IBM MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad de serie a un valor con el tipo de datos MQTYPE_FLOAT32 o MQTYPE_FLOAT64, la serie debe tener el formato siguiente:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

e_char
Un carácter exponente, que es "E" o "e".

e_sign
Un carácter de signo más (+) o signo menos (-) opcional para el exponente.

e_digits
Una secuencia contigua de caracteres de dígitos (0-9) para el exponente. Al menos, debe estar presente un carácter de dígito, si la serie contiene un carácter de exponente.

Detrás de la secuencia de caracteres de dígitos, o los caracteres opcionales que representan un exponente, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por supuesto que la serie representa un número de coma flotante decimal con un exponente que es una potencia de 10.

IBM MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad numérico en una serie, el valor se convierte a la representación de serie del valor como un número decimal, no la serie que contiene el carácter ASCII para ese valor. Por ejemplo, el entero 65 se convierte a la serie "65", no a la serie "A".

- Al convertir un valor de propiedad de serie de bytes en una serie, cada byte se convierte en los dos caracteres hexadecimales que representan el byte. Por ejemplo, la matriz de bytes {0xF1, 0x12, 0x00, 0xFF} se convierte a la serie "F11200FF".

MQIM_QUERY_LENGTH
Consulte el tipo y la longitud del valor de propiedad. La longitud se devuelve en el parámetro **DataLength** de la llamada MQINQMP. El valor de propiedad no se devuelve.

Si se especifica un almacenamiento intermedio **ReturnedName** , el campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El nombre de propiedad no se devuelve.

Opciones de iteración: Las opciones siguientes están relacionadas con la iteración sobre propiedades, utilizando un nombre con un carácter comodín

MQIM_INQ_PRIMERO
Consultar la primera propiedad que coincide con el nombre especificado. Después de esta llamada, se establece un cursor en la propiedad que se devuelve.

Éste es el valor predeterminado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP, si es necesario, para volver a consultar la misma propiedad.

Tenga en cuenta que sólo hay un cursor de propiedad; por lo tanto, si el nombre de propiedad, especificado en la llamada MQINQMP, cambia el cursor se restablece.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_NEXT
MQIM_INQ_PROP_UNDER_CURSOR

MQIM_INQ_NEXT
Consulta la siguiente propiedad que coincide con el nombre especificado, continuando la búsqueda desde el cursor de propiedad. El cursor está avanzado a la propiedad que se devuelve.

Si esta es la primera llamada MQINQMP para el nombre especificado, se devuelve la primera propiedad que coincide con el nombre especificado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP si es necesario, para volver a consultar la misma propiedad.

Si la propiedad bajo el cursor se ha suprimido, MQINQMP devuelve la siguiente propiedad coincidente después de la que se ha suprimido.

Si se añade una propiedad que coincide con el comodín, mientras una iteración está en curso, es posible que la propiedad se devuelva o no durante la finalización de la iteración. La propiedad se devuelve una vez que la iteración se reinicia utilizando MQIMPO_INQ_FIRST.

Una propiedad que coincide con el comodín que se ha suprimido, mientras la iteración estaba en curso, no se devuelve después de su supresión.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_PRIMERO
MQIM_INQ_PROP_UNDER_CURSOR

MQIM_INQ_PROP_UNDER_CURSOR
Recupere el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez, utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, cuando se especifica el descriptor de mensaje en el campo *MsgHandle* del MQGMO en una llamada MQGET, o cuando se especifica el descriptor de mensaje en los campos *OriginalMsgHandle* o *NewMsgHandle* de la estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando el cursor de propiedad todavía no se ha establecido, o si la propiedad a la que apunta el cursor de propiedad se ha suprimido, la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_PRIMERO
MQIM_INQ_NEXT

Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

MQIMPO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQIMPO_NONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_INQ_FIRST.

RequestedEncoding (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo RequestedEncoding

Esta es la codificación en la que se convertirá el valor de propiedad consultado cuando se especifique MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQENC_NATIVE.

RequestedCCSID (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo RequestedCCSID

El juego de caracteres en el que se va a convertir el valor de propiedad consultado si el valor es una serie de caracteres. También es el juego de caracteres en el que se convertirá *ReturnedName* cuando se especifique MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQCCSI_APPL.

ReturnedEncoding (MQLONG)
Estructura de opciones de propiedad de mensaje de consulta-Campo ReturnedEncoding

En la salida, es la codificación del valor devuelto.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el campo *ReturnedEncoding* , en el momento de la devolución, es el mismo valor que el valor que se ha pasado.

El valor inicial de este campo es MQENC_NATIVE.

ReturnedCCSID (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo ReturnedCCSID

En la salida, es el juego de caracteres del valor devuelto si el parámetro **Type** de la llamada MQINQMP es MQTYPE_STRING.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el campo *ReturnedCCSID* , en el momento de la devolución, es el mismo valor que el valor que se ha pasado.

El valor inicial de este campo es cero.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco (campo de 4 bytes).

ReturnedName (MQCHARV)

Consultar estructura de opciones de propiedad de mensaje-Campo ReturnedName

El nombre real de la propiedad consultado.

En la entrada, se puede pasar un almacenamiento intermedio de serie utilizando el campo *VSPtr* o *VSOffset* de la estructura MQCHARV . La longitud del almacenamiento intermedio de serie se especifica utilizando el campo *VSBufsize* de la estructura MQCHARV.

Al volver de la llamada MQINQMP, el almacenamiento intermedio de serie se completa con el nombre de la propiedad que se ha consultado, siempre que el almacenamiento intermedio de serie sea lo suficientemente largo como para contener completamente el nombre. El campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El campo *VSCCSID* de la estructura MQCHARV se rellena para indicar el juego de caracteres del nombre devuelto, si la conversión del nombre ha fallado o no.

Es un campo de entrada/salida. El valor inicial de este campo es MQCHARV_DEFAULT.

TypeString (MQCHAR8)

Consultar estructura de opciones de propiedad de mensaje-Campo TypeString

Una representación de serie del tipo de datos de la propiedad.

Si la propiedad se ha especificado en una cabecera MQRFH2 y no se reconoce el atributo MQRFH2 `dt` , este campo se puede utilizar para determinar el tipo de datos de la propiedad. *TypeString* se devuelve en el juego de caracteres codificado 1208 (UTF-8), y son los primeros ocho bytes del valor del atributo `dt` de la propiedad que no se ha podido reconocer

Siempre es un campo de salida. El valor inicial de este campo es la serie nula en el lenguaje de programación C y 8 caracteres en blanco en otros lenguajes de programación.

MQMD - Descriptor de mensaje

La estructura MQMD contiene la información de control que acompaña a los datos de aplicación cuando un mensaje viaja entre las aplicaciones de envío y recepción. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1 .

Disponibilidad

Todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQMD es MQMD_VERSION_2. Las aplicaciones que están pensadas para ser portables entre varios entornos deben asegurarse de que la versión necesaria de MQMD esté soportada en todos los entornos afectados. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQMD soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQMD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Hay disponible una declaración para la estructura version-1 con el nombre MQMD1.

Juego de caracteres y codificación

Los datos de MQMD deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Si los gestores de colas emisores y receptores utilizan conjuntos de caracteres o codificaciones diferentes, los datos de MQMD se convierten automáticamente. No es necesario que la aplicación convierta MQMD.

Utilización de distintas versiones de MQMD

Un MQMD de version-2 es equivalente a utilizar un MQMD de version-1 y a prefijar los datos del mensaje con una estructura MQMDE. Sin embargo, si todos los campos de la estructura MQMDE tienen sus valores predeterminados, se puede omitir MQMDE. Un MQMD version-1 más MQMDE se utilizan tal como se describe:

- En las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD version-1 , la aplicación puede opcionalmente añadir un prefijo a los datos de mensaje con un MQMDE, estableciendo el campo *Format* de MQMD en MQFMT_MD_EXTENSION para indicar que existe un MQMDE. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE.

  Nota: Varios de los campos que existen en MQMD version-2 pero no en MQMD version-1 son campos de entrada/salida en las llamadas MQPUT y MQPUT1 . Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .
- En la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. El campo *Format* de MQMD tendrá el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Los valores predeterminados que el gestor de colas utiliza para los campos de MQMDE son los mismos que los valores iniciales de dichos campos, que se muestran en la Tabla 503 en la página 476.

Cuando un mensaje está en una cola de transmisión, algunos de los campos de MQMD se establecen en valores determinados; consulte “MQXQH-Cabecera de cola de transmisión” en la página 625 para obtener más detalles.

Contexto de mensaje

Determinados campos de MQMD contienen el contexto del mensaje. Hay dos tipos de contexto de mensaje: *contexto de identidad* y *contexto de origen*. Por norma, se utiliza para:

- El contexto de identidad está relacionado con la aplicación que *originalmente* ha colocado el mensaje
- El contexto de origen está relacionado con la aplicación que *más recientemente* ha colocado el mensaje.

Estas dos aplicaciones pueden ser la misma aplicación, pero también pueden ser aplicaciones diferentes (por ejemplo, cuando se reenvía un mensaje de una aplicación a otra).

Aunque la identidad y el contexto de origen normalmente tienen los significados descritos, el contenido de ambos tipos de campos de contexto en MQMD depende de las opciones MQPMO_ * _CONTEXT que se especifican cuando se coloca el mensaje. Como resultado, el contexto de identidad no está necesariamente relacionado con la aplicación que colocó originalmente el mensaje, y el contexto de origen no está necesariamente relacionado con la aplicación que colocó el mensaje más recientemente; depende del diseño de la suite de aplicaciones.

El agente de canal de mensajes (MCA) nunca altera el contexto del mensaje. Los MCA que reciben mensajes de gestores de colas remotos utilizan la opción de contexto MQPMO_SET_ALL_CONTEXT en la llamada MQPUT o MQPUT1 . Esto permite al MCA receptor conservar exactamente el contexto de mensaje que ha viajado con el mensaje del MCA emisor. Sin embargo, el resultado es que el contexto

de origen no está relacionado con ninguno de los MCA que han enviado y recibido el mensaje. El contexto de origen hace referencia a una aplicación anterior que ha colocado el mensaje. Si todas las aplicaciones intermedias han pasado el contexto de mensaje, el contexto de origen hace referencia a la propia aplicación de origen.

En las descripciones, los campos de contexto se describen como si se utilizaran como se ha descrito anteriormente. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 500. Campos en MQMD para MQMD

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQMD_STRUC_ID	'MD'
Versión (número de versión de estructura)	MQMD_VERSION_1	1
Informe (opciones para mensajes de informe)	MQRO_NONE	0
MsgType (tipo de mensaje)	MQMT_DATAGRAM	8
MQMD-Campo de caducidad (duración del mensaje)	MQEI_UNLIMITED	-1
MQMD-Campo de comentarios (comentario o código de razón)	MQFB_NONE	0
Codificación (codificación numérica de datos de mensaje)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (identificador de juego de caracteres de datos de mensaje)	MQCCSI_Q_MGR	0
Formato (nombre de formato de datos de mensaje)	MQFMT_NONE	Espacios en blanco
Prioridad (prioridad de mensaje)	MQPRI_PRIORITY_AS_Q_DEF	-1
Persistence (persistencia de mensaje)	MQPER_PERSISTENCE_AS_Q_DEF	2
MQMD-Campo MsgId (identificador de mensaje)	MQMI_NONE	Nulos
CorrelId (identificador de correlación)	MQCI_NONE	Nulos
BackoutCount (contador de restituciones)	Ninguna	0
ReplyToQ (nombre de la cola de respuestas)	Ninguna	Serie nula o espacios en blanco
ReplyToQMgr (nombre del gestor de colas de respuestas)	Ninguna	Serie nula o espacios en blanco
UserIdentifier (identificador de usuario)	Ninguna	Serie nula o espacios en blanco
AccountingToken (señal de contabilidad)	MQACT_NONE	Nulos
Datos deApplIdentity (datos de aplicación relacionados con la identidad)	Ninguna	Serie nula o espacios en blanco

Tabla 500. Campos en MQMD para MQMD (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
TipoPutAppl (tipo de aplicación que ha colocado el mensaje)	MQAT_NO_CONTEXT	0
PutApplName (nombre de la aplicación que ha colocado el mensaje)	Ninguna	Serie nula o espacios en blanco
PutDate (fecha en la que se colocó el mensaje)	Ninguna	Serie nula o espacios en blanco
PutTime (hora a la que se colocó el mensaje)	Ninguna	Serie nula o espacios en blanco
Datos deApplOrigin (datos de aplicación relacionados con el origen)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Version* es menor que MQMD_VERSION_2.		
GroupId (identificador de grupo)	MQGI_NONE	Nulos
MsgSeqNúmero (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
Desplazamiento (desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0
MQMD-Campo MsgFlags (distintivos de mensaje)	MQMF_NONE	0
OriginalLength (longitud del mensaje original)	MQOL_UNDEFINED	-1

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQMD_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMD MyMD = {MQMD_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura y debe ser:

MQMD_STRUC_ID
Identificador de la estructura del descriptor de mensaje.

Para el lenguaje de programación C, también se define la constante MQMD_STRUC_ID_ARRAY; tiene el mismo valor que MQMD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQMD_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura y debe ser uno de los siguientes:

MQMD_VERSION_1
Estructura del descriptor de mensaje Version-1 .

Esta versión está soportada en todos los entornos.

MQMD_VERSION_2
Estructura del descriptor de mensaje Version-2 .

Esta versión está soportada en todos los entornos IBM MQ V6.0 y posteriores, además de IBM MQ MQI clients conectados a estos sistemas.

Nota: Cuando se utiliza un MQMD version-2 , el gestor de colas realiza comprobaciones adicionales en cualquier estructura de cabecera MQ que pueda estar presente al principio de los datos del mensaje de aplicación; para obtener más detalles, consulte las notas de uso de la llamada MQPUT.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQMD_CURRENT_VERSION
Versión actual de la estructura del descriptor de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQMD_VERSION_1.

Report (MQLONG)

Un mensaje de informe es un mensaje sobre otro mensaje, utilizado para informar a una aplicación sobre sucesos esperados o inesperados relacionados con el mensaje original. El campo *Report* permite a la aplicación que envía el mensaje original especificar qué mensajes de informe son necesarios, si los datos de mensaje de aplicación se van a incluir en ellos y también (para informes y respuestas) cómo se van a establecer los identificadores de mensaje y correlación en el informe o mensaje de respuesta. Se puede solicitar cualquiera o todos (o ninguno) de los siguientes tipos de mensajes de informe:

- Excepción
- Caducidad
- Confirmar al llegar (COA)
- Confirmar en entrega (COD)
- notificación de acción positiva (PAN)
- notificación de acción negativa (NAN)

Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

La aplicación que recibe el mensaje de informe puede determinar la razón por la que se ha generado el informe examinando el campo *Feedback* en el MQMD; consulte el campo *Feedback* para obtener más detalles.

El uso de opciones de informe al colocar un mensaje en un tema puede hacer que se generen y envíen a la aplicación cero, uno o varios mensajes de informe. Esto se debe a que el mensaje de publicación puede enviarse a cero, una o muchas aplicaciones de suscripción.

Opciones de excepción: especifique una de las opciones listadas para solicitar un mensaje de informe de excepción.

MQRO_EXCEPTION
Un agente de canal de mensajes genera este tipo de informe cuando se envía un mensaje a otro gestor de colas y el mensaje no se puede entregar a la cola de destino especificada. Por ejemplo, la cola de destino o una cola de transmisión intermedia puede estar llena, o el mensaje puede ser demasiado grande para la cola.

La generación del mensaje de informe de excepción depende de la persistencia del mensaje original y de la velocidad del canal de mensajes (normal o rápido) a través del cual viaja el mensaje original:

- Para todos los mensajes persistentes y para los mensajes no persistentes que viajan a través de canales de mensajes normales, el informe de excepción sólo se genera si la acción especificada por la aplicación emisora para la condición de error se puede completar correctamente. La aplicación emisora puede especificar una de las acciones siguientes para controlar la disposición del mensaje original cuando se produce la condición de error:

– MQRO_DEAD_LETTER_Q (coloca el mensaje original en la cola de mensajes no entregados).
– MQRO_DISCARD_MSG (descarta el mensaje original).

Si la acción especificada por la aplicación emisora no se puede completar correctamente, el mensaje original se deja en la cola de transmisión y no se genera ningún mensaje de informe de excepción.

- Para los mensajes no persistentes que viajan a través de canales de mensajes rápidos, el mensaje original se elimina de la cola de transmisión y el informe de excepción genera *incluso si* la acción especificada para la condición de error no se puede completar correctamente. Por ejemplo, si se especifica MQRO_DEAD_LETTER_Q, pero el mensaje original no se puede colocar en la cola de mensajes no entregados porque dicha cola está llena, se genera el mensaje de informe de excepción y se descarta el mensaje original.

  Para obtener más información sobre los canales de mensajes normales y rápidos, consulte Velocidad de mensajes no persistentes (NPMSPEED).

No se genera un informe de excepción si la aplicación que ha colocado el mensaje original puede recibir una notificación síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 .

Las aplicaciones también pueden enviar informes de excepción, para indicar que un mensaje no se puede procesar (por ejemplo, porque es una transacción de débito que haría que la cuenta excediera su límite de crédito).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA
Es lo mismo que MQRO_EXCEPTION, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA
Informes de excepción con datos completos necesarios.

Es lo mismo que MQRO_EXCEPTION, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

Opciones de caducidad: especifique una de las opciones listadas para solicitar un mensaje de informe de caducidad.

MQRO_EXPIRATION
Este tipo de informe lo genera el gestor de colas si el mensaje se descarta antes de la entrega a una aplicación porque ha pasado su hora de caducidad (consulte el campo *Expiry*). Si esta opción no está establecida, no se genera ningún mensaje de informe si se descarta un mensaje por esta razón (incluso si especifica una de las opciones MQRO_EXCEPTION_ *).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA
Es el mismo que MQRO_EXPIRATION, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original

contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA
Es lo mismo que MQRO_EXPIRATION, excepto que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

Opciones de confirmación al llegar: especifique una de las opciones listadas para solicitar un mensaje de informe de confirmación al llegar.

MQRO_COA

Este tipo de informe lo genera el gestor de colas propietario de la cola de destino cuando el mensaje se coloca en la cola de destino. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se coloca como parte de una unidad de trabajo y la cola de destino es una cola local, el mensaje de informe de COA generado por el gestor de colas sólo se puede recuperar si se confirma la unidad de trabajo.

No se genera un informe COA si el campo *Format* del descriptor de mensaje es MQFMT_XMIT_Q_HEADER o MQFMT_DEAD_LETTER_HEADER. Esto impide que se genere un informe de COA si el mensaje se coloca en una cola de transmisión o no se puede entregar y se coloca en una cola de mensajes no entregados.

En el caso de una cola puente IMS , el informe COA se genera cuando el mensaje llega a la cola IMS (acuse de recibo recibido de IMS) y no cuando el mensaje se coloca en la cola puente MQ . Esto significa que si IMS no está activo, no se genera ningún informe COA hasta que se inicia IMS y se pone en cola un mensaje en la cola IMS .

El usuario que ejecuta un programa que coloca un mensaje con MQMD.Report= MQRO_COA debe tener autorización + passid en la cola de respuestas. Si el usuario no tiene autorización + passid, el mensaje de informe COA no llega a la cola de respuestas. Se ha intentado colocar el mensaje de informe en la cola de mensajes no entregados.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_DATA
Es el mismo que MQRO_COA, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_FULL_DATA
Es lo mismo que MQRO_COA, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

Opciones de confirmación de entrega: especifique una de las opciones listadas para solicitar un mensaje de informe de confirmación de entrega.

MQRO_COD

Este tipo de informe lo genera el gestor de colas cuando una aplicación recupera el mensaje de la cola de destino de una forma que suprime el mensaje de la cola. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se recupera como parte de una unidad de trabajo, el mensaje de informe se genera dentro de la misma unidad de trabajo, de modo que el informe no está disponible hasta que se confirma la unidad de trabajo. Si la unidad de trabajo se restituye, el informe no se envía.

No siempre se genera un informe COD si se recupera un mensaje con la opción MQGMO_MARK_SKIP_BACKOUT. Si la unidad de trabajo primaria se restituye pero se confirma la unidad de trabajo secundaria, el mensaje se elimina de la cola, pero no se genera un informe COD.

No se genera un informe COD si el campo *Format* del descriptor de mensaje es MQFMT_DEAD_LETTER_HEADER. Esto impide que se genere un informe COD si el mensaje no se puede entregar y se coloca en una cola de mensajes no entregados.

MQRO_COD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_DATA

Es el mismo que MQRO_COD, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

Si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET para el mensaje original y el mensaje recuperado se trunca, la cantidad de datos de mensaje de aplicación colocados en el mensaje de informe depende del entorno:

- En z/OS, es el mínimo de:
  - La longitud del mensaje original
  - La longitud del almacenamiento intermedio utilizado para recuperar el mensaje
  - 100 bytes.
- En otros entornos, es el mínimo de:
  - La longitud del mensaje original
  - 100 bytes.

MQRO_COD_WITH_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_FULL_DATA

Es lo mismo que MQRO_COD, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

MQRO_COD_WITH_FULL_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

Opciones de notificación de acción: especifique una o ambas de las opciones listadas para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.

MQRO_PAN

Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

MQRO_NAN
Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje no se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe. Por ejemplo, es posible que desee incluir algunos datos que indiquen por qué no se ha podido realizar la solicitud.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

La solicitud debe determinar qué condiciones corresponden a una acción positiva y cuáles a una acción negativa. Sin embargo, si la solicitud sólo se ha realizado parcialmente, genere un informe NAN en lugar de un informe PAN si se solicita. Cada condición posible debe corresponder a una acción positiva, o a una acción negativa, pero no a ambas.

Opciones de identificador de mensaje: especifique una de las opciones listadas para controlar cómo se va a establecer el *MsgId* del mensaje de informe (o del mensaje de respuesta).

MQRO_NEW_MSG_ID
Esta es la acción predeterminada e indica que si se genera un informe o respuesta como resultado de este mensaje, se genera un nuevo *MsgId* para el informe o mensaje de respuesta.

MQRO_PASS_MSG_ID
Si se genera un informe o respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *MsgId* del informe o mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el informe o mensaje de respuesta será diferente para cada uno.

Si no se especifica esta opción, se presupone MQRO_NEW_MSG_ID.

Opciones de identificador de correlación: especifique una de las opciones listadas para controlar cómo se va a establecer el *CorrelId* del mensaje de informe (o del mensaje de respuesta).

MQRO_COPY_MSG_ID_TO_CORREL_ID
Esta es la acción predeterminada e indica que si se genera un informe o respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *CorrelId* del informe o mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el *CorrelId* del informe o mensaje de respuesta será diferente para cada uno.

MQRO_PASS_CORREL_ID
Si se genera un informe o respuesta como resultado de este mensaje, el *CorrelId* de este mensaje se copia en el *CorrelId* del informe o mensaje de respuesta.

El *CorrelId* de un mensaje de publicación será específico de un suscriptor a menos que utilice la opción MQSO_SET_CORREL_ID y establezca el campo de ID SubCorrelen MQSD en MQCI_NONE. Por lo tanto, es posible que el *CorrelId* copiado en el *CorrelId* del informe o mensaje de respuesta sea diferente para cada uno.

Si no se especifica esta opción, se presupone MQRO_COPY_MSG_ID_TO_CORREL_ID.

Los servidores que respondían a solicitudes o generaban mensajes de informe deben comprobar si las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID se habían establecido en el mensaje original. Si lo eran, los servidores deben realizar la acción descrita para estas opciones. Si no se establece ninguna, los servidores deben realizar la acción predeterminada correspondiente.

Opciones de disposición: especifique una de las opciones listadas para controlar la disposición del mensaje original cuando no se puede entregar a la cola de destino. La aplicación puede establecer las opciones de disposición independientemente de solicitar informes de excepción.

MQRO_DEAD_LETTER_Q
Es la acción predeterminada y coloca el mensaje en la cola de mensajes no entregados si el mensaje no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

MQRO_DISCARD_MSG
Esto descarta el mensaje si no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

Si desea devolver el mensaje original al remitente, sin que el mensaje original se coloque en la cola de mensajes no entregados, el remitente debe especificar MQRO_DISCARD_MSG con MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_AND_EXPIRY
Si esta opción se establece en un mensaje y se genera un informe o una respuesta debido a él, el descriptor de mensaje del informe hereda:

- MQRO_DISCARD_MSG si se ha establecido.
- El tiempo de caducidad restante del mensaje (si no es un informe de caducidad). Si se trata de un informe de caducidad, el tiempo de caducidad se establece en 60 segundos.

Opción de actividad

MQRO_ACTIVITY
El uso de este valor permite rastrear la ruta de **cualquier** mensaje a través de una red de gestores de colas. La opción de informe se puede especificar en cualquier mensaje de usuario actual, lo que permite al instante empezar a calcular la ruta del mensaje a través de la red.

Si la aplicación que genera el mensaje no puede habilitar la generación de informes de actividad, la creación de informes se puede habilitar utilizando una salida cruzada de API proporcionada por los administradores del gestor de colas.

Nota:

1. Cuanto menos sean los gestores de colas de la red capaces de generar informes de actividad, menos detallada será la ruta.
2. Los informes de actividad pueden ser difíciles de colocar en el orden correcto para determinar la ruta tomada.
3. Es posible que los informes de actividad no puedan encontrar una ruta a su destino solicitado.
4. Los mensajes con este conjunto de opciones de informe deben ser aceptados por cualquier gestor de colas, incluso si no entienden la opción. Esto permite que la opción de informe se establezca en cualquier mensaje de usuario, incluso si lo procesa un gestor de colas anterior aIBM WebSphere MQ 6.0 .
5. Si un proceso, ya sea un gestor de colas o un proceso de usuario, realiza una actividad en un mensaje con esta opción establecida, puede elegir generar y colocar un informe de actividad.

Opción predeterminada: especifique lo siguiente si no se necesitan opciones de informe:

MQRO_NONE

Utilice este valor para indicar que no se ha especificado ninguna otra opción. MQRO_NONE está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Información general:

1. La aplicación que envía el mensaje original debe solicitar específicamente todos los tipos de informe necesarios. Por ejemplo, si se solicita un informe de COA pero no se solicita un informe de excepción, se genera un informe de COA cuando el mensaje se coloca en la cola de destino, pero no se genera ningún informe de excepción si la cola de destino está llena cuando el mensaje llega allí. Si no se establece ninguna opción *Report* , el gestor de colas o el agente de canal de mensajes (MCA) no genera ningún mensaje de informe.

   Se pueden especificar algunas opciones de informe aunque el gestor de colas local no las reconozca; esto es útil cuando el gestor de colas de *destino* debe procesar la opción. Consulte "Opciones de informe y distintivos de mensaje" en la página 919 para obtener más detalles.

   Si se solicita un mensaje de informe, el nombre de la cola a la que enviar el informe debe especificarse en el campo *ReplyToQ* . Cuando se recibe un mensaje de informe, la naturaleza del informe se puede determinar examinando el campo *Feedback* en el descriptor de mensaje.
2. Si el gestor de colas o MCA que genera un mensaje de informe no puede colocar el mensaje de informe en la cola de respuestas (por ejemplo, porque la cola de respuestas o la cola de transmisión está llena), el mensaje de informe se coloca en su lugar en la cola de mensajes no entregados. Si *también* falla, o no hay ninguna cola de mensajes no entregados, la acción realizada depende del tipo de mensaje de informe:

   - Si el mensaje de informe es un informe de excepción, el mensaje que ha generado el informe de excepción se deja en su cola de transmisión; esto garantiza que el mensaje no se pierda.
   - Para todos los demás tipos de informe, el mensaje de informe se descarta y el proceso continúa normalmente. Esto se hace porque el mensaje original ya se ha entregado de forma segura (para los mensajes de informe COA o COD), o ya no es de ningún interés (para un mensaje de informe de caducidad).

   Una vez que un mensaje de informe se ha colocado correctamente en una cola (ya sea la cola de destino o una cola de transmisión intermedia), el mensaje ya no está sujeto a un proceso especial; se trata igual que cualquier otro mensaje.
3. Cuando se genera el informe, se abre la cola *ReplyToQ* y el mensaje de informe se coloca utilizando la autorización del *UserIdentifier* en el MQMD del mensaje que provoca el informe, excepto en los casos siguientes:

   - Los informes de excepción generados por un MCA receptor se colocan con la autorización que el MCA utilizó cuando intentó colocar el mensaje que causaba el informe.
   - Los informes de COA generados por el gestor de colas se colocan con la autorización que se haya utilizado cuando el mensaje que ha provocado el informe se ha colocado en el gestor de colas que genera el informe. Por ejemplo, si el mensaje ha sido transferido por un MCA receptor utilizando el identificador de usuario del MCA, el gestor de colas coloca el informe COA utilizando el identificador de usuario del MCA.

   Las aplicaciones que generan informes deben utilizar la misma autorización que utilizan para generar una respuesta; normalmente es la autorización del identificador de usuario en el mensaje original.

   Si el informe tiene que viajar a un destino remoto, los remitentes y receptores pueden decidir si lo aceptan, del mismo modo que lo hacen para otros mensajes.
4. Si se solicita un mensaje de informe con datos:

   - El mensaje de informe siempre se genera con la cantidad de datos solicitados por el remitente del mensaje original. Si el mensaje de informe es demasiado grande para la cola de respuestas, se produce el proceso descrito anteriormente; el mensaje de informe nunca se trunca para que quepa en la cola de respuestas.

- Si el *Format* del mensaje original es MQFMT_XMIT_Q_HEADER, los datos incluidos en el informe no incluyen MQXQH. Los datos del informe empiezan con el primer byte de los datos más allá de MQXQH en el mensaje original. Esto ocurre tanto si la cola es una cola de transmisión como si no.

5. Si se recibe un mensaje de informe de COA, COD o caducidad en la cola de respuestas, se garantiza que el mensaje original ha llegado, se ha entregado o ha caducado, según corresponda. Sin embargo, si se solicita uno o varios de estos mensajes de informe y no se recibe, no se puede suponer lo contrario, porque podría haberse producido una de las situaciones siguientes:

   a. El mensaje de informe se retiene porque un enlace está inactivo.

   b. El mensaje de informe se retiene porque existe una condición de bloqueo en una cola de transmisión intermedia o en la cola de respuestas (por ejemplo, la cola está llena o inhibida para colocaciones).

   c. El mensaje de informe está en una cola de mensajes no entregados.

   d. Cuando el gestor de colas intentaba generar el mensaje de informe, no podía colocarlo en la cola adecuada, ni en la cola de mensajes no entregados, por lo que no se podía generar el mensaje de informe.

   e. Se ha producido una anomalía del gestor de colas entre la acción que se notifica (llegada, entrega o caducidad) y la generación del mensaje de informe correspondiente. (Esto no sucede para los mensajes de informe COD si la aplicación recupera el mensaje original dentro de una unidad de trabajo, ya que el mensaje de informe COD se genera dentro de la misma unidad de trabajo.)

   Los mensajes de informe de excepción se pueden contener de la misma forma por las razones 1, 2 y 3 anteriores. Sin embargo, cuando un MCA no puede generar un mensaje de informe de excepción (el mensaje de informe no se puede colocar en la cola de respuestas o en la cola de mensajes no entregados), el mensaje original permanece en la cola de transmisión del emisor y el canal se cierra. Esto se produce independientemente de si el mensaje de informe se iba a generar en el extremo emisor o receptor del canal.

6. Si el mensaje original se bloquea temporalmente (lo que genera un mensaje de informe de excepción y el mensaje original se coloca en una cola de mensajes no entregados), pero el bloqueo se borra y una aplicación lee el mensaje original de la cola de mensajes no entregados y lo coloca de nuevo en su destino, puede ocurrir lo siguiente:

   - Aunque se ha generado un mensaje de informe de excepción, el mensaje original finalmente llega correctamente a su destino.
   - Se genera más de un mensaje de informe de excepción con respecto a un único mensaje original, porque el mensaje original puede encontrar otro bloqueo más adelante.

Informar de mensajes al transferir a un tema:

1. Se pueden generar informes al colocar un mensaje en un tema. Este mensaje se enviará a todos los suscriptores del tema, que podría ser cero, uno o muchos. Esto debe tenerse en cuenta al elegir utilizar las opciones de informe, ya que se podrían generar muchos mensajes de informe como resultado.
2. Al colocar un mensaje en un tema, puede haber muchas colas de destino a las que se debe dar una copia del mensaje. Si algunas de estas colas de destino tienen un problema, como la cola llena, la finalización satisfactoria de MQPUT depende del valor de NPMSGDLV o PMSGDLV (en función de la persistencia del mensaje). Si el valor es tal que la entrega de mensajes a la cola de destino debe ser satisfactoria (por ejemplo, es un mensaje persistente para un suscriptor duradero y PMSGDLV se establece en ALL o ALLDUR), el éxito se define como uno de los siguientes criterios que se cumplen:

   - Se ha colocado correctamente en la cola de suscriptores
   - Uso de MQRO_DEAD_LETTER_Q y una colocación satisfactoria en la cola de mensajes no entregados si la cola de suscriptores no puede tomar el mensaje
   - Utilice MQRO_DISCARD_MSG si la cola de suscriptores no puede tomar el mensaje.

Mensajes de informe para segmentos de mensajes:

1. Se pueden solicitar mensajes de informe para los mensajes que tienen la segmentación permitida (consulte la descripción del distintivo MQMF_SEGMENTATION_ALLOWED). Si el gestor de colas considera necesario segmentar el mensaje, se puede generar un mensaje de informe para cada uno de los segmentos que posteriormente encuentren la condición relevante. Las aplicaciones deben estar preparadas para recibir varios mensajes de informe para cada tipo de mensaje de informe solicitado. Utilice el campo *GroupId* en el mensaje de informe para correlacionar los varios informes con el identificador de grupo del mensaje original, y el campo *Feedback* identifique el tipo de cada mensaje de informe.
2. Si se utiliza MQGMO_LOGICAL_ORDER para recuperar mensajes de informe para segmentos, tenga en cuenta que las llamadas MQGET sucesivas pueden devolver informes de *tipos diferentes* . Por ejemplo, si se solicitan los informes COA y COD para un mensaje segmentado por el gestor de colas, las llamadas MQGET para los mensajes de informe pueden devolver los mensajes de informe COA y COD intercalados de forma imprevisible. Evite esto utilizando la opción MQGMO_COMPLETE_MSG (opcionalmente con MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG hace que el gestor de colas vuelva a ensamblar los mensajes de informe que tienen el mismo tipo de informe. Por ejemplo, la primera llamada MQGET podría volver a ensamblar todos los mensajes COA relacionados con el mensaje original, y la segunda llamada MQGET podría volver a ensamblar todos los mensajes COD. Lo que se reensambla primero depende del tipo de mensaje de informe que aparezca primero en la cola.
3. Las aplicaciones que ellas mismas colocan segmentos pueden especificar diferentes opciones de informe para cada segmento. Sin embargo, tenga en cuenta los siguientes puntos:
   - Si los segmentos se recuperan utilizando la opción MQGMO_COMPLETE_MSG, el gestor de colas solo respeta las opciones de informe del *primer* segmento.
   - Si los segmentos se recuperan uno por uno, y la mayoría de ellos tienen una de las opciones MQRO_COD_ *, pero al menos un segmento no lo hace, no puede utilizar la opción MQGMO_COMPLETE_MSG para recuperar los mensajes de informe con una sola llamada MQGET, o utilizar la opción MQGMO_ALL_SEGMENTS_AVAILABLE para detectar cuándo han llegado todos los mensajes de informe.
4. En una red MQ , los gestores de colas pueden tener distintas prestaciones. Si un mensaje de informe para un segmento es generado por un gestor de colas o MCA que no soporta la segmentación, el gestor de colas o MCA no incluye de forma predeterminada la información de segmento necesaria en el mensaje de informe, y esto podría dificultar la identificación del mensaje original que ha hecho que se generara el informe. Evite esta dificultad solicitando datos con el mensaje de informe, es decir, especificando las opciones MQRO_ * _WITH_DATA o MQRO_ * _WITH_FULL_DATA adecuadas. Sin embargo, tenga en cuenta que si se especifica MQRO_ * _WITH_DATA, es posible que se devuelvan *menos de* 100 bytes de datos de mensaje de aplicación a la aplicación que recupera el mensaje de informe, si el mensaje de informe lo genera un gestor de colas o un MCA que no soporta la segmentación.

Contenido del descriptor de mensaje para un mensaje de informe: cuando el gestor de colas o el agente de canal de mensajes (MCA) genera un mensaje de informe, establece los campos del descriptor de mensaje en los valores siguientes y, a continuación, coloca el mensaje de la forma normal.

Tabla 501. Valores utilizados para los campos MQMD cuando se genera un mensaje de informe generado por el sistema

Campo de MQMD	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	MQRO_NONE
MsgType	MQMT_REPORT
Expiry	MQEI_UNLIMITED
Feedback	Según corresponda para la naturaleza del informe (MQFB_COA, MQFB_COD, MQFB_EXPIRATION o un valor MQRC_ *)

Tabla 501. Valores utilizados para los campos MQMD cuando se genera un mensaje de informe generado por el sistema (continuación)

Campo de MQMD	Valor utilizado
Encoding	Copiado del descriptor de mensaje original
CodedCharSetId	Copiado del descriptor de mensaje original
Format	Copiado del descriptor de mensaje original
Priority	Copiado del descriptor de mensaje original
Persistence	Copiado del descriptor de mensaje original
MsgId	Según lo especificado por las opciones de informe en el descriptor de mensaje original
CorrelId	Según lo especificado por las opciones de informe en el descriptor de mensaje original
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Nombre de gestor de colas
UserIdentifier	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
AccountingToken	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
ApplIdentityData	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
PutApplType	MQAT_QMGR, o según corresponda para el agente de canal de mensajes
PutApplName	Primeros 28 bytes del nombre del gestor de colas o del nombre del agente de canal de mensajes. Para los mensajes de informe generados por el puente IMS , este campo contiene el nombre de grupo XCF y el nombre de miembro XCF del sistema IMS con el que se relaciona el mensaje.
PutDate	Fecha en la que se envía el mensaje de informe
PutTime	Hora a la que se envía el mensaje de informe
ApplOriginData	Espacios en blanco
GroupId	Copiado del descriptor de mensaje original
MsgSeqNumber	Copiado del descriptor de mensaje original
Offset	Copiado del descriptor de mensaje original
MsgFlags	Copiado del descriptor de mensaje original
OriginalLength	Copiado del descriptor de mensaje original si no es MQOL_UNDEFINED, y establecido en la longitud de los datos de mensaje originales de lo contrario

Se recomienda una aplicación que genere un informe para establecer valores similares, excepto para lo siguiente:

- El campo *ReplyToQMgr* se puede establecer en blancos (el gestor de colas lo cambia por el nombre del gestor de colas local cuando se coloca el mensaje).
- Establezca los campos de contexto utilizando la opción que se habría utilizado para una respuesta, normalmente MQPMO_PASS_IDENTITY_CONTEXT.

Análisis del campo de informe: el campo *Report* contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas en "Análisis del campo de informe" en la página 920.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQRO_NONE.

MsgType (MQLONG)

Indica el tipo del mensaje. Los tipos de mensaje se agrupan de la forma siguiente:

MQMT_SYSTEM_FIRST
Valor más bajo para tipos de mensajes definidos por el sistema.

MQMT_SYSTEM_LAST
Valor más alto para tipos de mensajes definidos por el sistema.

Los valores siguientes están definidos actualmente dentro del rango del sistema:

MQMT_DATAGRAM
El mensaje es uno que no requiere una respuesta.

MQMT_REQUEST
El mensaje es uno que requiere una respuesta.

Especifique el nombre de la cola a la que enviar la respuesta en el campo *ReplyToQ* . El campo *Report* indica cómo establecer *MsgId* y *CorrelId* de la respuesta.

MQMT_REPLY
El mensaje es la respuesta a un mensaje de solicitud anterior (MQMT_REQUEST). El mensaje debe enviarse a la cola indicada por el campo *ReplyToQ* del mensaje de solicitud. Utilice el campo *Report* de la solicitud para controlar cómo establecer *MsgId* y *CorrelId* de la respuesta.

Nota: El gestor de colas no impone la relación de solicitud-respuesta; esto es una responsabilidad de la aplicación.

MQMT_REPORT
El mensaje está informando sobre alguna aparición esperada o inesperada, normalmente relacionada con algún otro mensaje (por ejemplo, se ha recibido un mensaje de solicitud que contenía datos que no eran válidos). Envíe el mensaje a la cola indicada por el campo *ReplyToQ* del descriptor de mensaje del mensaje original. Establezca los campos *Feedback* para indicar la naturaleza del informe. Utilice el campo *Report* del mensaje original para controlar cómo establecer *MsgId* y *CorrelId* del mensaje de informe.

Los mensajes de informe generados por el gestor de colas o el agente de canal de mensajes siempre se envían a la cola *ReplyToQ* , con los campos *Feedback* y *CorrelId* establecidos como se ha descrito anteriormente.

También se pueden utilizar valores definidos por la aplicación. Deben estar dentro del rango siguiente:

MQMT_APPL_PRIMERO
Valor más bajo para tipos de mensajes definidos por la aplicación.

MQMT_APPL_LAST
Valor más alto para tipos de mensajes definidos por la aplicación.

Para las llamadas MQPUT y MQPUT1 , el valor *MsgType* debe estar dentro del rango definido por el sistema o del rango definido por la aplicación; si no lo está, la llamada falla con el código de razón MQRC_MSG_TYPE_ERROR.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQMT_DATAGRAM.

Expiry (MQLONG)

Es un periodo de tiempo expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. El mensaje se debe poder seleccionar para descartarlo si no se ha suprimido de la cola de destino antes de que transcurra este período de tiempo.

Por ejemplo, para establecer un minuto para la hora de caducidad, debe establecer **MQMD**.**Expiry** a 600.

El valor se reduce para reflejar el tiempo que el mensaje pasa en la cola de destino, y también en cualquier cola de transmisión intermedia si la transferencia es a una cola remota. También pueden disminuir los agentes de canal de mensajes para reflejar los tiempos de transmisión, si estos son significativos. Del mismo modo, una aplicación que reenvía este mensaje a otra cola puede disminuir el valor si es necesario, si ha retenido el mensaje durante un tiempo significativo. Sin embargo, la hora de caducidad se trata como aproximada y no es necesario reducir el valor para reflejar intervalos de tiempo pequeños.

Cuando una aplicación recupera el mensaje utilizando la llamada MQGET, el campo *Expiry* representa el tiempo de caducidad que todavía permanece.

Una vez transcurrido el tiempo de caducidad de un mensaje, éste pasa a ser elegible para ser descartado por el gestor de colas. El mensaje se descarta cuando se produce una llamada MQGET de examinar o no examinar que habría devuelto el mensaje si no hubiera caducado. Por ejemplo, una llamada MQGET sin examinar con el campo *MatchOptions* en MQGMO establecido en MQMO_NONE leyendo de una cola ordenada FIFO descarta todos los mensajes caducados hasta el primer mensaje no caducado. Con una cola ordenada de prioridad, la misma llamada descartará los mensajes caducados de prioridad más alta y los mensajes de prioridad igual que han llegado a la cola antes del primer mensaje no caducado.

Un mensaje que ha caducado nunca se devuelve a una aplicación (mediante una llamada MQGET de examinar o no examinar), por lo que el valor del campo *Expiry* del descriptor de mensaje después de una llamada MQGET satisfactoria es mayor que cero o el valor especial MQEI_UNLIMITED.

Si un mensaje se coloca en una cola remota, el mensaje puede caducar (y descartarse) mientras está en una cola de transmisión intermedia, antes de que el mensaje llegue a la cola de destino.

Se genera un informe cuando se descarta un mensaje caducado, si el mensaje ha especificado una de las opciones de informe MQRO_EXPIRATION_ *. Si no se especifica ninguna de estas opciones, no se genera ningún informe de este tipo; se presupone que el mensaje ya no es relevante después de este periodo de tiempo (quizás porque un mensaje posterior lo ha reemplazado).

Para un mensaje colocado en un punto de sincronización, el intervalo de caducidad empieza en el momento en que se coloca el mensaje, no en el momento en que se confirma el punto de sincronización. Es posible que el intervalo de caducidad pueda pasar antes de que se confirme el punto de sincronización. En este caso, el mensaje se descartará en algún momento después de la operación de confirmación y el mensaje no se devolverá a una aplicación en respuesta a una operación MQGET.

Cualquier otro programa que descarte mensajes basándose en la hora de caducidad también debe enviar un mensaje de informe adecuado si se ha solicitado uno.

Notas:

1. Si se coloca un mensaje con una hora *Expiry* de cero o un número mayor que 999 999 999, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_EXPIRY_ERROR; en este caso no se genera ningún mensaje de informe.

   Para habilitar el código de razón 2013, MQRC_EXPIRY_ERROR, debe habilitar la variable de entorno AMQ_ENFORCE_MAX_EXPIRY_ERROR.

   A continuación se utiliza un ejemplo para Linux:

   ```
   $ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
   ```

   Tenga en cuenta que:

   - Lo importante es exportar la variable
   - El valor real se ignora, sin embargo, el uso de `True` puede ser útil al revisar la configuración.
2. Debido a que un mensaje con un tiempo de caducidad que ha transcurrido puede que no se descarte hasta más adelante, es posible que haya mensajes en una cola que hayan pasado su tiempo de caducidad y que, por lo tanto, no sean elegibles para la recuperación. No obstante, estos mensajes cuentan para el número de mensajes en la cola para todos los fines, incluido el desencadenamiento de profundidad.

Si un suscriptor/consumidor (cliente) intenta obtener un mensaje y dicho mensaje ha caducado, el cliente no recibe nada porque el mensaje se ha descartado porque era demasiado antiguo. Además, el cliente no recibirá ningún mensaje de error.

3. Se genera un informe de caducidad, si se solicita, cuando se descarta el mensaje, no cuando pasa a ser elegible para descartarlo.
4. Descartar un mensaje caducado y generar un informe de caducidad si se solicita, nunca forman parte de la unidad de trabajo de la aplicación, aunque el mensaje se haya planificado para descartarlo como resultado de una llamada MQGET que opera dentro de una unidad de trabajo.
5. Si una llamada MQGET recupera un mensaje casi caducado dentro de una unidad de trabajo y posteriormente se restituye la unidad de trabajo, es posible que el mensaje sea apto para ser descartado antes de que se pueda recuperar de nuevo.
6. Si un mensaje casi caducado está bloqueado por una llamada MQGET con MQGMO_LOCK, es posible que el mensaje sea apto para ser descartado antes de que pueda ser recuperado por una llamada MQGET con MQGMO_MSG_UNDER_CURSOR; el código de razón MQRC_NO_MSG_UNDER_CURSOR se devuelve en esta llamada MQGET posterior si esto sucede.
7. Cuando se recupera un mensaje de solicitud con un tiempo de caducidad mayor que cero, la aplicación puede realizar una de las acciones siguientes cuando envía el mensaje de respuesta:
   - Copie el tiempo de caducidad restante del mensaje de solicitud en el mensaje de respuesta.
   - Establezca la hora de caducidad en el mensaje de respuesta en un valor explícito mayor que cero.
   - Establezca la hora de caducidad en el mensaje de respuesta en MQEI_UNLIMITED.

   La acción a realizar depende del diseño de la aplicación. Sin embargo, la acción predeterminada para transferir mensajes a una cola de mensajes no entregados (undelivered-message) debe ser conservar el tiempo de caducidad restante del mensaje y continuar decrementándolo.
8. Los mensajes desencadenantes siempre se generan con MQEI_UNLIMITED.
9. Un mensaje (normalmente en una cola de transmisión) que tiene un nombre *Format* de MQFMT_XMIT_Q_HEADER tiene un segundo descriptor de mensaje dentro de MQXQH. Por lo tanto, tiene dos campos *Expiry* asociados. En este caso deben señalarse los siguientes puntos adicionales:
   - Cuando una aplicación coloca un mensaje en una cola remota, el gestor de colas coloca el mensaje inicialmente en una cola de transmisión local y añade un prefijo a los datos del mensaje de aplicación con una estructura MQXQH. El gestor de colas establece los valores de los dos campos *Expiry* para que sean los mismos que los especificados por la aplicación.

     Si una aplicación coloca un mensaje directamente en una cola de transmisión local, los datos del mensaje ya deben empezar con una estructura MQXQH y el nombre de formato debe ser MQFMT_XMIT_Q_HEADER. En este caso, no es necesario que la aplicación establezca los valores de estos dos campos *Expiry* para que sean iguales. (El gestor de colas comprueba que el campo *Expiry* de MQXQH contiene un valor válido y que los datos del mensaje son lo suficientemente largos para incluirlo). Para una aplicación que puede escribir directamente en la cola de transmisión, la aplicación tiene que crear una cabecera de cola de transmisión con el descriptor de mensaje incorporado. Sin embargo, si el valor de caducidad del descriptor de mensaje grabado en la cola de transmisión no es coherente con el valor del descriptor de mensaje incorporado, se produce un rechazo de error de caducidad.
   - Cuando un mensaje con un nombre *Format* de MQFMT_XMIT_Q_HEADER se recupera de una cola (ya sea una cola normal o de transmisión), el gestor de colas disminuye *ambos* estos *Expiry* campos con el tiempo empleado en esperar en la cola. No se genera ningún error si los datos del mensaje no son lo suficientemente largos para incluir el campo *Expiry* en MQXQH.
   - El gestor de colas utiliza el campo *Expiry* en el descriptor de mensaje independiente (es decir, no el del descriptor de mensaje incorporado en la estructura MQXQH) para probar si el mensaje es apto para descartarlo.
   - Si los valores iniciales de los dos campos *Expiry* son diferentes, la hora *Expiry* en el descriptor de mensaje separado cuando se recupera el mensaje puede ser mayor que cero (por lo que el

mensaje no es elegible para descartarlo), mientras que ha transcurrido la hora según el campo *Expiry* en MQXQH. En este caso, el campo *Expiry* en MQXQH se establece en cero.

10. La hora de caducidad de un mensaje de respuesta devuelto desde el puente IMS es ilimitada a menos que se establezca MQIIH_PASS_EXPIRATION en el campo Distintivos de MQIIH. Consulte Distintivos para obtener más información.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED
El mensaje tiene un tiempo de caducidad ilimitado.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQEI_UNLIMITED.

Mensajes caducados en z/OS
En IBM MQ for z/OS, los mensajes que han caducado se descartan mediante la siguiente llamada MQGET adecuada.

Sin embargo, si no se produce ninguna llamada de este tipo, el mensaje caducado no se descarta y, para algunas colas, se puede acumular un gran número de mensajes caducados. Para remediarlo, establezca el gestor de colas para explorar las colas periódicamente y descartar los mensajes caducados en una o más colas de una de las siguientes maneras:

Exploración periódica
Puede especificar un periodo utilizando el atributo de gestor de colas EXPRYINT (intervalo de caducidad). Cada vez que se alcanza el intervalo de caducidad, el gestor de colas busca las colas candidatas que vale la pena explorar para descartar los mensajes caducados.

El gestor de colas mantiene información sobre los mensajes caducados en cada cola y sabe si vale la pena realizar una exploración de los mensajes caducados. Por lo tanto, sólo se explora una selección de colas en cualquier momento.

Sólo un gestor de colas de un grupo de compartición de colas explora las colas compartidas. Generalmente, es el primer gestor de colas que se reinicia o el primero que tiene establecido EXPRYINT. Si este gestor de colas termina, otro gestor de colas del grupo de compartición de colas se hace cargo de la exploración de colas. Establezca el valor de intervalo de caducidad para todos los gestores de colas de un grupo de compartición de colas en el mismo valor.

Tenga en cuenta que el proceso de caducidad tiene lugar para cada cola cuando se reinicia un gestor de colas, independientemente del valor EXPRYINT.

Solicitud explícita
Emita el mandato REFRESH QMGR TYPE (EXPIRE), especificando la cola o colas que desea explorar.

Imposición de tiempos de caducidad más bajos
Los administradores pueden limitar el tiempo de caducidad de cualquier mensaje transferido a una cola o un tema utilizando el atributo **CAPEXPRY** especificado en el atributo **CUSTOM** en la cola o el tema.

Una hora de caducidad especificada en el campo **Expiry** del MQMD, por una aplicación, que es mayor que el valor **CAPEXPRY** especificado en el atributo **CUSTOM** en la cola o tema se sustituirá por ese valor **CAPEXPRY** . Se utilizará una hora de caducidad especificada por una aplicación, que es inferior al valor **CAPEXPRY** .

Tenga en cuenta que el valor de **CAPEXPRY** se expresa en décimas de segundos, por lo que un minuto tiene un valor de 600.

Si se utiliza más de un objeto en la vía de acceso de resolución, por ejemplo, cuando se coloca un mensaje en un alias o en una cola remota, se utiliza el valor más bajo de todos los valores **CAPEXPRY** como límite superior para la caducidad del mensaje.

Los cambios en los valores de **CAPEXPRY** entran en vigor inmediatamente. El valor de caducidad se evalúa para cada colocación en una cola o tema y, por lo tanto, es sensible a la resolución de objetos, que puede diferir entre cada operación de colocación.

Sin embargo, tenga en cuenta que los mensajes existentes en la cola, antes de un cambio en **CAPEXPRY**, no se ven afectados por el cambio (es decir, su tiempo de caducidad permanece intacto). Sólo los mensajes nuevos que se colocan en la cola después del cambio en **CAPEXPRY** tienen la nueva hora de caducidad.

Por ejemplo, en un clúster donde se realiza una transferencia a una cola abierta con MQOO_BIND_NOT_FIXED, a los mensajes se les pueden asignar distintos valores de caducidad en cada transferencia, en función del valor **CAPEXPRY** establecido para la cola de transmisión, utilizado por el canal, que envía el mensaje al gestor de colas de destino seleccionado.

Tenga en cuenta que una transferencia a una cola o tema por parte de una aplicación JMS que especifica un retardo de entrega falla con MQRC_EXPIRY_ERROR, si el retardo de entrega supera el tiempo de caducidad resuelto para la cola o tema de destino. Un atributo **CAPEXPRY** establecido en una cola resuelta para un destino JMS puede provocar este error.

Nota: CAPEXPRY no se debe utilizar en ninguna cola que contenga mensajes generados internamente por IBM MQ como, por ejemplo, cualquier SYSTEM.CLUSTER.* y SYSTEM.PROTECTION.POLICY.QUEUE.

Referencia relacionada

DEFINE colas

DEFINE TOPIC

Feedback (MQLONG)

El campo Comentarios se utiliza con un mensaje de tipo MQMT_REPORT para indicar la naturaleza del informe y sólo es significativo con ese tipo de mensaje.

El campo puede contener uno de los valores MQFB_ * o uno de los valores MQRC_ *. Los códigos de comentarios se agrupan de la forma siguiente:

MQFB_NONE
No se han proporcionado comentarios.

MQFB_SISTEMA_PRIMERO
Valor más bajo para los comentarios generados por el sistema.

MQFB_SYSTEM_LAST
Valor más alto para comentarios generados por el sistema.

El rango de códigos de comentarios generados por el sistema MQFB_SYSTEM_FIRST a MQFB_SYSTEM_LAST incluye los códigos de comentarios generales listados en este tema (MQFB_ *), y también los códigos de razón (MQRC_ *) que se pueden producir cuando el mensaje no se puede colocar en la cola de destino.

MQFB_APPL_FIRST
Valor más bajo para los comentarios generados por la aplicación.

MQFB_APPL_LAST
Valor más alto para los comentarios generados por la aplicación.

Las aplicaciones que generan mensajes de informe no deben utilizar códigos de comentarios en el rango del sistema (que no sean MQFB_QUIT), a menos que deseen simular mensajes de informe generados por el gestor de colas o el agente de canal de mensajes.

En las llamadas MQPUT o MQPUT1 , el valor especificado debe ser MQFB_NONE o estar dentro del rango del sistema o del rango de aplicaciones. Esto se comprueba independientemente del valor de *MsgType*.

Códigos de comentarios generales:

MQFB_COA
Confirmación de llegada a la cola de destino (consulte MQRO_COA).

MQFB_COD
Confirmación de entrega a la aplicación receptora (consulte MQRO_COD).

MQFB_EXPIRATION
El mensaje se ha descartado porque no se había eliminado de la cola de destino antes de que hubiera transcurrido su tiempo de caducidad.

MQFB_PAN
Notificación de acción positiva (consulte MQRO_PAN).

MQFB_NAN
Notificación de acción negativa (consulte MQRO_NAN).

MQFB_QUIT
Finalizar aplicación.

Esto lo puede utilizar un programa de planificación de carga de trabajo para controlar el número de instancias de un programa de aplicación que se están ejecutando. El envío de un mensaje MQMT_REPORT con este código de retorno a una instancia del programa de aplicación indica a dicha instancia que debe detener el proceso. Sin embargo, el cumplimiento de este convenio es un asunto de la aplicación; el gestor de colas no lo impone.

Códigos de comentarios de canal:

MQFB_CHANNEL_COMPLETED
Un canal ha finalizado normalmente.

MQFB_CHANNEL_FAIL
Un canal ha finalizado de forma anómala y pasa al estado STOPPED.

MQFB_CHANNEL_FAIL_RETRY
Un canal ha finalizado de forma anómala y entra en estado RETRY.

Códigos de comentarios deIMS-bridge

Estos códigos se utilizan cuando se recibe un código de detección IMS-OTMA inesperado. El código de detección o, cuando el código de detección es 0x1A , el código de razón asociado con ese código de detección, se indica en *Comentarios*.

1. Para los códigos *Feedback* en el rango MQFB_IMS_FIRST (300) a MQFB_IMS_LAST (399), se ha recibido un código de detección distinto de 0x1A . El *código de detección* lo proporciona la expresión (*Feedback* - MQFB_IMS_FIRST+1)
2. Para los códigos *Feedback* en el rango MQFB_IMS_NACK_1A_REASON_FIRST (600) a MQFB_IMS_NACK_1A_REASON_LAST (855), se ha recibido un código de detección de 0x1A . El *código de razón* asociado con el código de detección lo proporciona la expresión (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST)

El significado de los códigos de detección IMS-OTMA y los códigos de razón correspondientes se describen en la publicación *Open Transaction Manager Access Guide and Reference*.

El puente IMS puede generar los siguientes códigos de comentarios:

MQFB_DATA_LENGTH_ZERO
Una longitud de segmento era cero en los datos de aplicación del mensaje.

MQFB_DATA_LENGTH_NEGATIVO
Una longitud de segmento era negativa en los datos de aplicación del mensaje.

MQFB_DATA_LENGTH_TOO_BIG
Una longitud de segmento era demasiado grande en los datos de aplicación del mensaje.

MQFB_BUFFER_OVERFLOW
El valor de uno de los campos de longitud haría que los datos desbordaran el almacenamiento intermedio de mensajes.

MQFB_LENGTH_OFF_BY_ONE
El valor de uno de los campos de longitud era 1 byte demasiado corto.

MQFB_IIH_ERROR
El campo *Format* en MQMD especifica MQFMT_IMS, pero el mensaje no empieza por una estructura MQIIH válida.

MQFB_NOT_AUTHORIZED_FOR_IMS
El ID de usuario contenido en el descriptor de mensaje MQMD, o la contraseña contenida en el campo *Authenticator* de la estructura MQIIH, ha fallado la validación realizada por el puente IMS . Como resultado, el mensaje no se ha pasado a IMS.

MQFB_IMS_ERROR
IMSha devuelto un error inesperado. Consulte el registro de errores de IBM MQ en el sistema en el que reside el puente IMS para obtener más información sobre el error.

MQFB_IMS_FIRST
Cuando el código de detección IMS-OTMA no es 0x1A, los códigos de comentarios generados por IMSestán en el rango de MQFB_IMS_FIRST (300) a MQFB_IMS_LAST (399). El propio código de detección IMS-OTMA es *Feedback* menos MQFB_IMS_ERROR.

MQFB_IMS_LAST
Valor más alto para los comentarios generados por IMScuando el código de detección no es 0x1A.

MQFB_IMS_NACK_1A_REASON_FIRST
Cuando el código de detección es 0x1A, los códigos de comentarios generados por IMSestán en el rango de MQFB_IMS_NACK_1A_REASON_FIRST (600) a MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST
Valor más alto para los comentarios generados por IMScuando el código de detección es 0x1A

CICS-bridge feedback codes: CICS bridgepuede generar los siguientes códigos de comentarios:

MQFB_CICS_APPL_ABENDED
El programa de aplicación especificado en el mensaje ha finalizado de forma anómala. Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_APPL_NOT_STARTED
El EXEC CICS LINK para el programa de aplicación especificado en el mensaje ha fallado. Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_BRIDGE_FAILURE
CICS bridge ha terminado de forma anómala sin completar el proceso de errores normal.

MQFB_CICS_ERROR _CCSID
Identificador de juego de caracteres no válido.

MQFB_CICS_ERROR _CIH
Falta la estructura de cabecera de información de CICS o no es válida.

MQFB_CICS_COMMAREA_ERROR
La longitud de CICS COMMAREA no es válida.

MQFB_CICS_ID_CORREL_ERROR
Identificador de correlación no válido.

MQFB_CICS_ERROR _DLQ
La tarea CICS bridge no ha podido copiar una respuesta a esta solicitud en la cola de mensajes no entregados. La solicitud se ha restituido.

MQFB_CICS_ERROR _ENCODING_
Codificación no válida.

MQFB_CICS_ERROR DE INTERNAL
CICS bridge ha encontrado un error inesperado.

Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_NO_AUTORIZADO
Identificador de usuario no autorizado o contraseña no válida.

Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_UOW_BACKED_OUT
La unidad de trabajo fue restituido, por una de las siguientes razones:

- Se ha detectado una anomalía al procesar otra solicitud dentro de la misma unidad de trabajo.
- Se ha producido una terminación anómala de CICS mientras la unidad de trabajo estaba en curso.

MQFB_CICS_ERROR DE UOW
Campo de control de unidad de trabajo *UOWControl* no válido.

Códigos de retorno de mensaje de ruta de rastreo:

MQFB_ACTIVITY
Se utiliza con el formato MQFMT_EMBEDDED_PCF para permitir la opción de datos de usuario después de los informes de actividad.

MQFB_MAX_ACTIVITIES
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque el número de actividades en las que se ha implicado el mensaje supera el límite máximo de actividades.

MQFB_NO_REENVIADO
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque está a punto de enviarse a un gestor de colas remoto que no soporta mensajes de ruta de rastreo.

MQFB_NO_ENTREGADO
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque está a punto de colocarse en una cola local.

MQFB_UNSUPPORTED_FORWARDING
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque un valor del parámetro de reenvío no se reconoce y está en la máscara de bits rechazada.

MQFB_UNSUPPORTED_DELIVERY
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque un valor del parámetro de entrega no se reconoce y está en la máscara de bits rechazada.

IBM MQ códigos de razón: para los mensajes de informe de excepción, *Feedback* contiene un código de razón IBM MQ . Entre los posibles códigos de razón están:

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

Para obtener una lista completa de códigos de razón, consulte:

- Para IBM MQ for z/OS, consulte Códigos de terminación y razón de la API.
- Para todas las demás plataformas, consulte Códigos de terminación y razón de la API.

.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQFB_NONE.

Encoding (MQLONG)

Especifica la codificación numérica de los datos numéricos del mensaje; no se aplica a los datos numéricos de la propia estructura MQMD. La codificación numérica define la representación utilizada para enteros binarios, enteros decimales empaquetados y números de coma flotante.

En z/OS, la parte entera binaria del campo `Encoding` también se utiliza para especificar la codificación entera de los datos de caracteres en el cuerpo del mensaje cuando el identificador de juego de caracteres correspondiente indica que la representación del juego de caracteres depende de la codificación utilizada para los enteros binarios. Esto sólo afecta a determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Se define el siguiente valor especial:

MQENC_NATIVE
La codificación es el valor predeterminado para el lenguaje de programación y la máquina en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del lenguaje de programación y del entorno. Por este motivo, las aplicaciones deben compilarse utilizando los archivos de cabecera, macro, COPY o INCLUDE adecuados para el entorno en el que se ejecutará la aplicación.

Las aplicaciones que colocan mensajes suelen especificar MQENC_NATIVE. Las aplicaciones que recuperan mensajes deben comparar este campo con el valor MQENC_NATIVE; si los valores difieren, es posible que la aplicación tenga que convertir datos numéricos en el mensaje. Utilice la opción MQGMO_CONVERT para solicitar al gestor de colas que convierta el mensaje como parte del proceso de la llamada MQGET. Consulte "Codificaciones de máquina" en la página 916 para obtener detalles sobre cómo se construye el campo `Encoding` .

Si especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es la codificación a la que convertir los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica. Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa la codificación del mensaje no convertido que se devuelve a la aplicación.

En otros casos, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Este campo especifica el identificador de juego de caracteres de los datos de caracteres dentro del cuerpo del mensaje.

Nota: Los datos de tipo carácter en MQMD y las otras estructuras de datos de MQ que son parámetros en las llamadas deben estar en el juego de caracteres del gestor de colas. Esto lo define el atributo **`CodedCharSetId`** del gestor de colas; consulte "Atributos para el gestor de colas" en la página 810 para obtener detalles de este atributo.

Si este campo se establece en MQCCSI_Q_MGR al llamar a MQGET con MQGMO_CONVERT en las opciones, el comportamiento es diferente entre las aplicaciones cliente y servidor. Para las aplicaciones de servidor, la página de códigos utilizada para la conversión de caracteres es el `CodedCharSetId` del gestor de colas; para las aplicaciones cliente, la página de códigos utilizada para la conversión de caracteres es la página de códigos del entorno local actual.

Para las aplicaciones cliente, se rellena MQCCSI_Q_MGR, basándose en el entorno local del cliente en lugar del del gestor de colas. La excepción a esa regla es cuando coloca un mensaje en una cola de puente IMS ; lo que se devuelve, en el campo *`CodedCharSetId`* de MQMD, es el CCSID del gestor de colas.

No debe utilizar el siguiente valor especial:

MQCCSI_APPL
Esto da como resultado un valor incorrecto en el campo `CodedCharSetId` del MQMD y provoca un código de retorno de MQRC_SOURCE_CCSID_ERROR (o MQRC_FORMAT_ERROR para z/OS) cuando se recibe el mensaje utilizando la llamada MQGET con la opción MQGMO_CONVERT.

Puede utilizar los siguientes valores especiales:

MQCCSI_Q_MGR
Los datos de caracteres del mensaje están en el juego de caracteres del gestor de colas.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres verdadero del gestor de colas. Como resultado, la llamada MQGET nunca devuelve el valor MQCCSI_Q_MGR.

MQCCSI_PREDETERMINADO
El `CodedCharSetId` de los datos en el campo *`String`* está definido por el campo `CodedCharSetId` en la estructura de cabecera que precede a la estructura MQCFH, o por el campo `CodedCharSetId` en el MQMD si el MQCFH está al principio del mensaje.

MQCCSI_INHERIT
Los datos de caracteres del mensaje están en el mismo juego de caracteres que esta estructura; este es el juego de caracteres del gestor de colas. (Sólo para MQMD, MQCCSI_INHERIT tiene el mismo significado que MQCCSI_Q_MGR).

El gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres real de MQMD. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del campo `PutApplType` en MQMD es MQAT_BROKER.

MQCCSI_EMBEDDED
Los datos de tipo carácter del mensaje están en un juego de caracteres con el identificador contenido en los propios datos del mensaje. Puede haber cualquier número de identificadores de juego de caracteres incluidos en los datos del mensaje, que se aplican a diferentes partes de los datos. Este valor debe utilizarse para mensajes PCF (con un formato de MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF) que contengan datos en una mezcla de juegos de caracteres. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener especificado un identificador de juego de caracteres explícito y no MQCCSI_DEFAULT.

Si un mensaje con el formato MQFMT_EMBEDDED_PCF va a contener datos en una mezcla de juegos de caracteres, no utilice MQCCSI_EMBEDDED. En su lugar, establezca MQEPH_CCSID_EMBEDDED en el campo Distintivos de la estructura MQEPH. Esto es equivalente a establecer MQCCSI_EMBEDDED en la estructura anterior. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener especificado un identificador de juego de caracteres explícito y no MQCCSI_DEFAULT. Para obtener más información sobre la estructura MQEPH, consulte “MQEPH - Cabecera PCF incrustada” en la página 362.

Especifique este valor sólo en las llamadas MQPUT y MQPUT1 . Si se especifica en la llamada MQGET, impide la conversión del mensaje.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia los valores MQCCSI_Q_MGR y MQCCSI_INHERIT en el MQMD que se envía con el mensaje tal como se ha descrito anteriormente, pero no cambia el MQMD especificado en la llamada MQPUT o MQPUT1 . No se realiza ninguna otra comprobación en el valor especificado.

Las aplicaciones que recuperan mensajes deben comparar este campo con el valor que espera la aplicación; si los valores difieren, es posible que la aplicación tenga que convertir datos de tipo carácter en el mensaje.

En z/OS, el campo `Encoding` del MQMD se utiliza para especificar la codificación de enteros de datos de caracteres en el cuerpo del mensaje, cuando el campo `CodedCharSetId` del MQMD indica que la representación del juego de caracteres depende de la codificación utilizada para los enteros binarios. En Multiplatforms, se presupone que el orden de bytes de los datos de caracteres es el mismo que el de la codificación de enteros nativa para la plataforma donde se ejecuta el gestor de colas. Esto sólo afecta a determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16).

Si especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es el identificador del juego de caracteres codificado al que convertir los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica (excepto que el valor MQCCSI_Q_MGR o MQCCSI_INHERIT se convierte en el valor real). Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa el identificador de juego de caracteres codificado del mensaje no convertido que se devuelve a la aplicación.

De lo contrario, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQCCSI_Q_MGR.

Format (MQCHAR8)

Es un nombre que el remitente del mensaje utiliza para indicar al destinatario la naturaleza de los datos del mensaje. Los caracteres que estén en el juego de caracteres del gestor de colas se pueden especificar para el nombre, pero debe restringir el nombre a lo siguiente:

- Mayúsculas de la A a la Z
- Dígitos numéricos del 0 al 9

Si se utilizan otros caracteres, es posible que no sea posible convertir el nombre entre los juegos de caracteres de los gestores de colas emisor y receptor.

Rellene el nombre con espacios en blanco hasta la longitud del campo, o utilice un carácter nulo para terminar el nombre antes del final del campo; el nulo y los caracteres subsiguientes se tratan como blancos. No especifique un nombre con espacios en blanco iniciales o intercalados. Para la llamada MQGET, el gestor de colas devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

El gestor de colas no comprueba que el nombre cumpla con las recomendaciones descritas anteriormente.

Los nombres que empiezan por MQ en mayúsculas, minúsculas y combinación de mayúsculas y minúsculas tienen significados definidos por el gestor de colas; no utilice nombres que empiecen por estas letras para sus propios formatos. Los formatos incorporados del gestor de colas son:

MQFMT_NONE
La naturaleza de los datos no está definida: los datos no se pueden convertir cuando el mensaje se recupera de una cola utilizando la opción MQGMO_CONVERT.

Si especifica MQGMO_CONVERT en la llamada MQGET, y el juego de caracteres o la codificación de datos del mensaje difiere del especificado en el parámetro **MsgDesc** , el mensaje se devuelve con los siguientes códigos de terminación y razón (suponiendo que no haya otros errores):

- Código de terminación MQCC_WARNING y código de razón MQRC_FORMAT_ERROR si los datos MQFMT_NONE están al principio del mensaje.
- Código de terminación MQCC_OK y código de razón MQRC_NONE si los datos MQFMT_NONE están al final del mensaje (es decir, precedidos por una o más estructuras de cabecera MQ). Las estructuras de cabecera MQ se convierten al juego de caracteres solicitado y a la codificación en este caso.

Para el lenguaje de programación C, la constante MQFMT_NONE_ARRAY también está definida; tiene el mismo valor que MQFMT_NONE, pero es una matriz de caracteres en lugar de una serie.

MQFMT_ADMIN
El mensaje es un mensaje de petición o respuesta de servidor de mandatos en formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte Utilización de formatos de mandatos programables para obtener más información sobre la utilización de mensajes de formato de mandatos programables.

Para el lenguaje de programación C, también se define la constante MQFMT_ADMIN_ARRAY; tiene el mismo valor que MQFMT_ADMIN, pero es una matriz de caracteres en lugar de una serie.

MQFMT_CICS
Los datos del mensaje empiezan por la cabecera de información MQCIH de CICS , seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo `Format` en la estructura MQCIH.

z/OS En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir mensajes que tengan el formato MQFMT_CICS.

Para el lenguaje de programación C, la constante MQFMT_CICS_ARRAY también está definida; tiene el mismo valor que MQFMT_CICS, pero es una matriz de caracteres en lugar de una serie.

MQFMT_COMMAND_1
El mensaje es un mensaje de respuesta de servidor de mandatos MQSC que contiene el recuento de objetos, el código de terminación y el código de razón. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_COMMAND_1_ARRAY ; tiene el mismo valor que MQFMT_COMMAND_1, pero es una matriz de caracteres en lugar de una serie.

MQFMT_COMMAND_2
El mensaje es un mensaje de respuesta del servidor de mandatos MQSC que contiene información sobre los objetos solicitados. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_COMMAND_2_ARRAY ; tiene el mismo valor que MQFMT_COMMAND_2, pero es una matriz de caracteres en lugar de una serie.

MQFMT_DEAD_LETTER_HEADER
Los datos del mensaje empiezan por la cabecera MQDLH de mensaje no entregado. Los datos del mensaje original siguen inmediatamente la estructura MQDLH. El nombre de formato de los datos de mensaje originales lo proporciona el campo `Format` en la estructura MQDLH; consulte "MQDLH - Cabecera de mensajes no entregados" en la página 349 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Los informes COA y COD no se generan para los mensajes que tienen un `Format` de MQFMT_DEAD_LETTER_HEADER.

Para el lenguaje de programación C, también se define la constante MQFMT_DEAD_LETTER_HEADER_ARRAY; tiene el mismo valor que MQFMT_DEAD_LETTER_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_DIST_HEADER
Los datos del mensaje empiezan con la cabecera de lista de distribución MQDH; esto incluye las matrices de registros MQOR y MQPMR. La cabecera de lista de distribución puede ir seguida de datos adicionales. El formato de los datos adicionales (si los hay) se proporciona mediante el campo *`Format`* en la estructura MQDH; consulte "MQDH - Cabecera de distribución" en la página 343 para obtener detalles de esta estructura. Los mensajes con el formato MQFMT_DIST_HEADER se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_DIST_HEADER_ARRAY; tiene el mismo valor que MQFMT_DIST_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_EMBEDDED_PCF
Formato para un mensaje de ruta de rastreo, siempre que el valor del mandato PCF se establezca en MQCMD_TRACE_ROUTE. El uso de este formato permite que los datos de usuario se envíen junto con

el mensaje de ruta de rastreo, siempre que sus aplicaciones puedan hacer frente a los parámetros PCF anteriores.

La cabecera PCF debe ser la primera cabecera, o el mensaje no se tratará como un mensaje de ruta de rastreo. Esto significa que el mensaje no puede estar en un grupo y que los mensajes de ruta de rastreo no se pueden segmentar. Si se envía un mensaje de ruta de rastreo en un grupo, el mensaje se rechaza con el código de razón MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Tenga en cuenta que MQFMT_ADMIN también se puede utilizar para el formato de un mensaje de ruta de rastreo, pero en este caso no se pueden enviar datos de usuario junto con el mensaje de ruta de rastreo.

MQFMT_EVENT

El mensaje es un mensaje de suceso de MQ que notifica un suceso que se ha producido. Los mensajes de sucesos tienen la misma estructura que los mandatos programables; consulte Mensajes de mandatos PCF para obtener más información sobre esta estructura y Supervisión de sucesos para obtener información sobre sucesos.

Los mensajes de suceso Version-1 se pueden convertir en todos los entornos si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Los mensajes de suceso Version-2 sólo se pueden convertir en z/OS.

Para el lenguaje de programación C, la constante MQFMT_EVENT_ARRAY también está definida; tiene el mismo valor que MQFMT_EVENT, pero es una matriz de caracteres en lugar de una serie.

MQFMT_IMS

Los datos del mensaje empiezan con la cabecera de información MQIIH de IMS , que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo `Format` en la estructura MQIIH.

Para obtener detalles sobre cómo se maneja la estructura MQIIH cuando se utiliza MQGET con MQGMO_CONVERT, consulte “Format (MQCHAR8)” en la página 410 y “Formato ReplyTo(MQCHAR8)” en la página 411.

Para el lenguaje de programación C, la constante MQFMT_IMS_ARRAY también está definida; tiene el mismo valor que MQFMT_IMS, pero es una matriz de caracteres en lugar de una serie.

MQFMT_IMS_VAR_STRING

El mensaje es una serie de variable IMS , que es una serie con el formato `ll`*zzccc*, donde:

`ll`

es un campo de longitud de 2 bytes que especifica la longitud total del elemento de serie de variable IMS . Esta longitud es igual a la longitud de `ll` (2 bytes), más la longitud de *zz* (2 bytes), más la longitud de la propia serie de caracteres. `ll` es un entero binario de 2 bytes en la codificación especificada por el campo `Encoding` .

zz

es un campo de 2 bytes que contiene distintivos que son significativos para IMS. *zz* es una serie de bytes que consta de dos campos MQBYTE y se transmite sin cambiar de remitente a destinatario (es decir, *zz* no está sujeto a ninguna conversión).

ccc

es una serie de caracteres de longitud variable que contiene `ll-4` caracteres. *ccc* está en el juego de caracteres especificado por el campo `CodedCharSetId` .

En z/OS, los datos del mensaje pueden consistir en una secuencia de series de variables de IMS a tope, con cada serie con el formato `llzzccc`. No debe haber bytes omitidos entre series de variables IMS sucesivas. Esto significa que si la primera serie tiene una longitud impar, la segunda serie estará desalineada, es decir, no empezará en un límite que sea un múltiplo de dos. Tenga cuidado al construir estas series en máquinas que requieren alineación de tipos de datos elementales.

Utilice la opción MQGMO_CONVERT en la llamada MQGET para convertir mensajes que tengan el formato MQFMT_IMS_VAR_STRING.

Para el lenguaje de programación C, también se define la constante MQFMT_IMS_VAR_STRING_ARRAY; tiene el mismo valor que MQFMT_IMS_VAR_STRING, pero es una matriz de caracteres en lugar de una serie.

MQFMT_MD_EXTENSION

Los datos del mensaje empiezan con la extensión de descriptor de mensaje MQMDE y, opcionalmente, van seguidos de otros datos (normalmente, los datos del mensaje de aplicación). El nombre de formato, el juego de caracteres y la codificación de los datos que siguen a MQMDE se proporcionan mediante los campos `Format`, `CodedCharSetIdy` `Encoding` en MQMDE. Consulte "MQMDE-Extensión de descriptor de mensaje" en la página 474 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, la constante MQFMT_MD_EXTENSION_ARRAY también está definida; tiene el mismo valor que MQFMT_MD_EXTENSION, pero es una matriz de caracteres en lugar de una serie.

MQFMT_PCF

El mensaje es un mensaje definido por el usuario que se ajusta a la estructura de un mensaje de formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte Utilización de formatos de mandatos programables para obtener más información sobre la utilización de mensajes de formato de mandatos programables.

Para el lenguaje de programación C, la constante MQFMT_PCF_ARRAY también está definida; tiene el mismo valor que MQFMT_PCF, pero es una matriz de caracteres en lugar de una serie.

MQFMT_REF_MSG_HEADER

Los datos del mensaje empiezan con la cabecera de mensaje de referencia MQRMH y, opcionalmente, van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos se proporcionan mediante los campos `Format`, `CodedCharSetIdy` `Encoding` en MQRMH. Consulte "MQRMH - Cabecera de mensaje de referencia" en la página 554 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_REF_MSG_HEADER_ARRAY; tiene el mismo valor que MQFMT_REF_MSG_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_RF_HEADER

Los datos del mensaje empiezan con las reglas y la cabecera de formato MQRFH, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos (si los hay) se proporcionan mediante los campos `Format`, `CodedCharSetIdy` `Encoding` en MQRFH. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_RF_HEADER_ARRAY; tiene el mismo valor que MQFMT_RF_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_RF_HEADER_2
Los datos del mensaje empiezan con las reglas version-2 y la cabecera de formato MQRFH2, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos opcionales (si los hay) se proporcionan mediante los campos `Format`, `CodedCharSetIdy` `Encoding` en MQRFH2. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_RF_HEADER_2_ARRAY ; tiene el mismo valor que MQFMT_RF_HEADER_2, pero es una matriz de caracteres en lugar de una serie.

MQFMT_STRING
Los datos del mensaje de aplicación pueden ser una serie SBCS (juego de caracteres de un solo byte) o una serie DBCS (juego de caracteres de doble byte). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_STRING_ARRAY; tiene el mismo valor que MQFMT_STRING, pero es una matriz de caracteres en lugar de una serie.

MQFMT_TRIGGER
El mensaje es un mensaje desencadenante, descrito por la estructura MQTM; consulte "MQTM-Mensaje de desencadenante" en la página 606 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, la constante MQFMT_TRIGGER_ARRAY también está definida; tiene el mismo valor que MQFMT_TRIGGER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_WORK_INFO_HEADER
Los datos del mensaje empiezan con la cabecera de información de trabajo MQWIH, que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo `Format` en la estructura MQWIH.

z/OS En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir los datos de usuario en mensajes que tengan el formato MQFMT_WORK_INFO_HEADER. Sin embargo, la propia estructura MQWIH siempre se devuelve en el juego de caracteres y la codificación del gestor de colas (es decir, la estructura MQWIH se convierte tanto si se especifica la opción MQGMO_CONVERT como si no).

Para el lenguaje de programación C, también se define la constante MQFMT_WORK_INFO_HEADER_ARRAY; tiene el mismo valor que MQFMT_WORK_INFO_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_XMIT_Q_HEADER
Los datos del mensaje empiezan por la cabecera de cola de transmisión MQXQH. Los datos del mensaje original siguen inmediatamente a la estructura MQXQH. El nombre de formato de los datos de mensaje originales lo proporciona el campo `Format` de la estructura MQMD, que forma parte de la cabecera de cola de transmisión MQXQH. Consulte "MQXQH-Cabecera de cola de transmisión" en la página 625 para obtener detalles de esta estructura.

Los informes COA y COD no se generan para los mensajes que tienen un `Format` de MQFMT_XMIT_Q_HEADER.

Para el lenguaje de programación C, también se define la constante MQFMT_XMIT_Q_HEADER_ARRAY; tiene el mismo valor que MQFMT_XMIT_Q_HEADER, pero es una matriz de caracteres en lugar de una serie.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Priority (MQLONG)

Para las llamadas MQPUT y MQPUT1 , el valor debe ser mayor o igual que cero; cero es la prioridad más baja. También se puede utilizar el siguiente valor especial:

MQPRI_PRIORITY_AS_Q_DEF

- Si la cola es una cola de clúster, la prioridad del mensaje se toma del atributo **DefPriority** definido en el gestor de colas de *destino* que es propietario de la instancia concreta de la cola en la que se coloca el mensaje.

  Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también ¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?

  El valor de *DefPriority* se copia en el campo *Priority* cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia *DefPriority* , los mensajes que ya se han colocado en la cola no se verán afectados.
- Si la cola no es una cola de clúster, la prioridad del mensaje se toma del atributo **DefPriority** definido en el gestor de colas *local* , incluso si el gestor de colas de destino es remoto.

  Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso. Este puede ser:

  – una cola alias
  – Una cola local
  – Una definición local de una cola remota
  – Un alias de gestor de colas
  – Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

  El valor de *DefPriority* se copia en el campo *Priority* cuando se coloca el mensaje. Si posteriormente se cambia *DefPriority* , los mensajes que ya se han colocado no se verán afectados.

El valor devuelto por la llamada MQGET es siempre mayor o igual que cero; el valor MQPRI_PRIORITY_AS_Q_DEF nunca se devuelve.

Si un mensaje se coloca con una prioridad mayor que el máximo soportado por el gestor de colas local (este máximo lo proporciona el atributo de gestor de colas **MaxPriority**), el mensaje lo acepta el gestor de colas, pero se coloca en la cola con la prioridad máxima del gestor de colas; la llamada MQPUT o MQPUT1 se completa con MQCC_WARNING y el código de razón MQRC_PRIORITY_EXCEEDS_MAXIMUM. Sin embargo, el campo *Priority* conserva el valor especificado por la aplicación que ha colocado el mensaje.

En z/OS, si un mensaje con un número MsgSeqde 1 se coloca en una cola que tiene una secuencia de entrega de mensajes de MQMDS_PRIORITY y un tipo de índice de MQIT_GROUP_ID, la cola podría tratar el mensaje con una prioridad diferente. Si el mensaje se ha colocado en la cola con una prioridad de 0 o 1, se procesa como si tuviera una prioridad de 2. Esto se debe a que el orden de los mensajes colocados en este tipo de cola se ha optimizado para habilitar las pruebas de compleción de grupo eficientes. Para obtener más información sobre la secuencia de entrega de mensajes MQMDS_PRIORITY y el tipo de índice MQIT_GROUP_ID, consulte Atributo de secuenciaMsgDelivery.

Al responder a un mensaje, las aplicaciones deben utilizar la prioridad del mensaje de solicitud para el mensaje de respuesta. En otras situaciones, especificar MQPRI_PRIORITY_AS_Q_DEF permite que el ajuste de prioridad se lleve a cabo sin cambiar la aplicación.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQPRI_PRIORITY_AS_Q_DEF.

Persistence (MQLONG)

Esto indica si el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Para las llamadas MQPUT y MQPUT1 , el valor debe ser uno de los siguientes:

MQPER_PERSISTENT

El mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Una vez que se ha colocado el mensaje y se ha confirmado la unidad de trabajo en la que se ha colocado (si el mensaje se coloca como parte de una unidad de trabajo), el mensaje se conserva en el almacenamiento auxiliar. Permanece allí hasta que el mensaje se elimina de la cola y la unidad de trabajo en la que se obtuvo se ha confirmado (si el mensaje se recupera como parte de una unidad de trabajo).

Cuando se envía un mensaje persistente a una cola remota, un mecanismo de almacenamiento y reenvío mantiene el mensaje en cada gestor de colas a lo largo de la ruta al destino, hasta que se sepa que el mensaje ha llegado al siguiente gestor de colas.

Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas que se correlacionan con un objeto CFSTRUCT en CFLEVEL (2) o inferior, o donde el objeto CFSTRUCT se define como RECOVER (NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

MQPER_NOT_PERSISTENT

El mensaje no suele sobrevivir a anomalías del sistema o a reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar cuando se reinicia el gestor de colas.

En el caso de las colas NPMCLASS (HIGH), los mensajes no persistentes sobreviven a una conclusión y reinicio normales del gestor de colas.

En el caso de las colas compartidas, los mensajes no persistentes sobreviven a los reinicios del gestor de colas en el grupo de compartición de colas, pero no sobreviven a las anomalías del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

MQPER_PERSISTENCE_AS_Q_DEF

- Si la cola es una cola de clúster, la persistencia del mensaje se toma del atributo **DefPersistence** definido en el gestor de colas de *destino* que es propietario de la instancia concreta de la cola en la que se coloca el mensaje.

  Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también ¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?

  El valor de *DefPersistence* se copia en el campo *Persistence* cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia *DefPersistence* , los mensajes que ya se han colocado en la cola no se verán afectados.

- Si la cola no es una cola de clúster, la persistencia del mensaje se toma del atributo **DefPersistence** definido en el gestor de colas *local* , incluso si el gestor de colas de destino es remoto.

  Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso. Este puede ser:

  – una cola alias
  – Una cola local
  – Una definición local de una cola remota
  – Un alias de gestor de colas
  – Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

El valor de *DefPersistence* se copia en el campo *Persistence* cuando se coloca el mensaje. Si posteriormente se cambia *DefPersistence* , los mensajes que ya se han colocado no se verán afectados.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Al responder a un mensaje, las aplicaciones deben utilizar la persistencia del mensaje de solicitud para el mensaje de respuesta.

Para una llamada MQGET, el valor devuelto es MQPER_PERSISTENT o MQPER_NOT_PERSISTENT.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQPER_PERSISTENCE_AS_Q_DEF.

MsgId (MQBYTE24)

Es una serie de bytes que se utiliza para distinguir un mensaje de otro. Generalmente, ningún mensaje debe tener el mismo identificador de mensaje, aunque el gestor de colas no lo permita. El identificador de mensaje es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de mensaje es una serie de bytes y no una serie de caracteres, el identificador de mensaje no se convierte entre juegos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , si la aplicación especifica MQMI_NONE o MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje exclusivo[3]cuando se transfiera el mensaje y lo coloque en el descriptor de mensaje enviado con el mensaje. El gestor de colas también devuelve este identificador de mensaje en el descriptor de mensaje que pertenece a la aplicación emisora. La aplicación puede utilizar este valor para registrar información sobre mensajes concretos y para responder a consultas de otras partes de la aplicación.

Si el mensaje se está colocando en un tema, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario para cada mensaje publicado. Si la aplicación especifica MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje exclusivo para devolver en la salida. Si la aplicación especifica MQMI_NONE, el valor del campo *MsgId* en MQMD no se modifica al devolver la llamada.

Consulte la descripción de MQPMO_RETAIN en “Opciones de MQPMO (MQLONG)” en la página 508 para obtener más detalles sobre las publicaciones retenidas.

Si el mensaje se está colocando en una lista de distribución, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario, pero el valor del campo *MsgId* en MQMD no se modifica al devolver la llamada, incluso si se ha especificado MQMI_NONE o MQPMO_NEW_MSG_ID. Si la aplicación necesita conocer los identificadores de mensaje generados por el gestor de colas, la aplicación debe proporcionar registros MQPMR que contengan el campo *MsgId* .

La aplicación emisora también puede especificar un valor para el identificador de mensaje que no sea MQMI_NONE; esto detiene el gestor de colas que genera un identificador de mensaje exclusivo. Una aplicación que reenvía un mensaje puede utilizarlo para propagar el identificador de mensaje del mensaje original.

El gestor de colas no utiliza este campo excepto para:

---

[3] Un *MsgId* generado por el gestor de colas consta de un identificador de producto de 4 bytes (AMQ ¬ o CSQ ¬ in ASCII o EBCDIC, donde ¬ representa un carácter en blanco), seguido de una implementación específica del producto de una serie exclusiva. En IBM MQ , contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para asegurarse de que los identificadores de mensaje sean exclusivos. La capacidad de generar una serie exclusiva también depende de que el reloj del sistema no se cambie hacia atrás. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, la aplicación debe evitar generar identificadores con caracteres iniciales en el rango de A a I en ASCII o EBCDIC (de X'41 'a X'49' y de X'C1'a X'C9'). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales en estos rangos.

- Generar un valor exclusivo si se solicita, tal como se ha descrito anteriormente
- Entregue el valor a la aplicación que emite la solicitud de obtención para el mensaje
- Copie el valor en el campo *CorrelId* de cualquier mensaje de informe que genere sobre este mensaje (en función de las opciones de *Report*)

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo *MsgId* de la forma especificada por el campo *Report* del mensaje original, MQRO_NEW_MSG_ID o MQRO_PASS_MSG_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *MsgId* es uno de los cinco campos que se pueden utilizar para recuperar un mensaje determinado de la cola. Normalmente, la llamada MQGET devuelve el siguiente mensaje en la cola, pero se puede obtener un mensaje determinado especificando uno o más de los cinco criterios de selección, en cualquier combinación; estos campos son:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

La aplicación establece uno o más de estos campos en los valores necesarios y, a continuación, establece las opciones de coincidencia MQMO_ * correspondientes en el campo *MatchOptions* en MQGMO para utilizar estos campos como criterios de selección. Sólo los mensajes que tienen los valores especificados en esos campos son candidatos para la recuperación. El valor predeterminado para el campo *MatchOptions* (si no lo modifica la aplicación) es que coincida con el identificador de mensaje y el identificador de correlación.

En z/OS, los criterios de selección que puede utilizar están restringidos por el tipo de índice utilizado para la cola. Consulte el atributo de cola **IndexType** para obtener más detalles.

Normalmente, el mensaje devuelto es el *primer* mensaje de la cola que cumple los criterios de selección. Pero si se especifica MQGMO_BROWSE_NEXT, el mensaje devuelto es el *siguiente* mensaje que satisface los criterios de selección; la exploración de este mensaje se inicia con el mensaje *siguiente* la posición actual del cursor.

Nota: La cola se explora secuencialmente en busca de un mensaje que satisfaga los criterios de selección, por lo que los tiempos de recuperación son más lentos que si no se especifica ningún criterio de selección, especialmente si se tienen que explorar muchos mensajes antes de que se encuentre uno adecuado. Las excepciones a esto son:

- Multi una llamada MQGET de *CorrelId* en Multiplatforms de 64 bits donde el índice *CorrelId* elimina la necesidad de realizar una exploración secuencial verdadera.
- z/OS una llamada MQGET de *IndexType* en z/OS.

En ambos casos, se mejora el rendimiento de recuperación.

Consulte Tabla 495 en la página 392 para obtener más información sobre cómo se utilizan los criterios de selección en diversas situaciones.

Especificar MQMI_NONE como identificador de mensaje tiene el mismo efecto que no especificar MQMO_MATCH_MSG_ID, es decir, *cualquier* coincidencia de identificador de mensaje.

Este campo se ignora si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el parámetro **GetMsgOpts** de la llamada MQGET.

Al volver de una llamada MQGET, el campo *MsgId* se establece en el identificador de mensaje del mensaje devuelto (si lo hay).

Se puede utilizar el siguiente valor especial:

MQMI_NONE
No se ha especificado ningún identificador de mensaje.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQMI_NONE_ARRAY también está definida; tiene el mismo valor que MQMI_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada/salida para las llamadas MQGET, MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_MSG_ID_LENGTH. El valor inicial de este campo es MQMI_NONE.

CorrelId (MQBYTE24)

El campo CorrelId es una propiedad de la cabecera de mensaje que se puede utilizar para identificar un mensaje específico o un grupo de mensajes.

Es una serie de bytes que la aplicación puede utilizar para relacionar un mensaje con otro, o para relacionar el mensaje con otro trabajo que la aplicación está realizando. El identificador de correlación es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de correlación es una serie de bytes y no una serie de caracteres, el identificador de correlación no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , la aplicación puede especificar cualquier valor. El gestor de colas transmite este valor con el mensaje y lo entrega a la aplicación que emite la solicitud get para el mensaje.

Si la aplicación especifica MQPMO_NEW_CORREL_ID, el gestor de colas genera un identificador de correlación exclusivo que se envía con el mensaje y también se devuelve a la aplicación emisora en la salida de la llamada MQPUT o MQPUT1 .

Un identificador de correlación generado por el gestor de colas consta de un identificador de producto de 3 bytes (AMQ o CSQ en ASCII o EBCDIC), seguido de un byte reservado y una implementación específica del producto de una serie exclusiva. En IBM MQ , esta serie de implementación específica del producto contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres para asegurarse de que los identificadores de mensaje son exclusivos. La capacidad de generar una serie exclusiva también depende de que el reloj del sistema no se cambie hacia atrás. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, la aplicación debe evitar generar identificadores con caracteres iniciales en el rango de A a I en ASCII o EBCDIC (de X'41 'a X'49' y de X'C1'a X'C9'). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales en estos rangos.

Este identificador de correlación generado se conserva con el mensaje si se conserva y se utiliza como identificador de correlación cuando el mensaje se envía como publicación a los suscriptores que especifican MQCI_NONE en el campo de ID SubCorrelen el MQSD pasado en la llamada MQSUB. Consulte Opciones de MQPMO para obtener más detalles sobre las publicaciones retenidas.

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo *CorrelId* de la forma especificada por el campo *Report* del mensaje original, MQRO_COPY_MSG_ID_TO_CORREL_ID o MQRO_PASS_CORREL_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *CorrelId* es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje determinado para recuperarlo de la cola. Consulte la descripción del campo *MsgId* para obtener detalles sobre cómo especificar valores para este campo.

Especificar MQCI_NONE como identificador de correlación tiene el mismo efecto que no especificar MQMO_MATCH_CORREL_ID, es decir, *cualquier* identificador de correlación coincidirá.

Si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el parámetro **GetMsgOpts** de la llamada MQGET, este campo se ignora.

Al volver de una llamada MQGET, el campo *CorrelId* se establece en el identificador de correlación del mensaje devuelto (si lo hay).

Se pueden utilizar los siguientes valores especiales:

MQCI_NONE
No se ha especificado ningún identificador de correlación.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCI_NONE_ARRAY; tiene el mismo valor que MQCI_NONE, pero es una matriz de caracteres en lugar de una serie.

MQCI_SESIÓN_NUEVA
El mensaje es el inicio de una nueva sesión.

Este valor es reconocido por CICS bridge como indica el inicio de una nueva sesión, es decir, el inicio de una nueva secuencia de mensajes.

Para el lenguaje de programación C, la constante MQCI_NEW_SESSION_ARRAY también está definida; tiene el mismo valor que MQCI_NEW_SESSION, pero es una matriz de caracteres en lugar de una serie.

Para la llamada MQGET, es un campo de entrada/salida. Para las llamadas MQPUT y MQPUT1 , es un campo de entrada si no se especifica MQPMO_NEW_CORREL_ID y un campo de salida si se especifica MQPMO_NEW_CORREL_ID. La longitud de este campo la proporciona MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Nota:

No puede pasar el identificador de correlación de una publicación en una jerarquía. El campo lo utiliza el gestor de colas.

BackoutCount (MQLONG)

Es un recuento del número de veces que la llamada MQGET ha devuelto previamente el mensaje como parte de una unidad de trabajo y, posteriormente, se ha restituido. Ayuda a la aplicación a detectar errores de proceso que se basan en el contenido del mensaje. El recuento excluye las llamadas MQGET que especifican cualquiera de las opciones MQGMO_BROWSE_ *.

La precisión de este recuento se ve afectada por el atributo de cola **HardenGetBackout** ; consulte “Atributos para colas” en la página 848.

En z/OS, un valor de 255 significa que el mensaje se ha restituido 255 o más veces; el valor devuelto nunca es mayor que 255.

Es un campo de salida para la llamada MQGET. Se ignora para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es 0.

ReplyToQ (MQCHAR48)

Es el nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud get para el mensaje envía los mensajes MQMT_REPLY y MQMT_REPORT. El nombre es el nombre local de una cola definida en el gestor de colas identificado por *ReplyToQMgr*. Esta cola no debe ser una cola modelo, aunque el gestor de colas emisor no lo verifique cuando se transfiere el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo no debe estar en blanco si el campo *MsgType* tiene el valor MQMT_REQUEST, o si el campo *Report* solicita algún mensaje de informe. Sin embargo, el valor especificado (o sustituido) se pasa a la aplicación que emite la solicitud get para el mensaje, sea cual sea el tipo de mensaje.

Si el campo *ReplyToQMgr* está en blanco, el gestor de colas local busca el nombre *ReplyToQ* en sus propias definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor *ReplyToQ* del mensaje transmitido se sustituye por el valor del atributo **RemoteQName** de la definición de la cola remota, y este valor se devuelve en el descriptor de mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, *ReplyToQ* no se modifica.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para las colas; esto también es cierto para el nombre transmitido, si el *ReplyToQ* se sustituye en el mensaje transmitido. La única comprobación realizada es que se ha especificado un nombre, si las circunstancias lo requieren.

Si no es necesaria una cola de respuestas, establezca el campo *ReplyToQ* en blancos, o (en el lenguaje de programación C) en la serie nula, o en uno o más blancos seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Si un mensaje que requiere un mensaje de informe no se puede entregar, y el mensaje de informe tampoco se puede entregar a la cola especificada, tanto el mensaje original como el mensaje de informe van a la cola de mensajes no entregados (undelivered-message) (consulte el atributo **DeadLetterQName** descrito en "Atributos para el gestor de colas" en la página 810).

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ReplyToGestor de colas (MQCHAR48)

Es el nombre del gestor de colas al que enviar el mensaje de respuesta o el mensaje de informe. *ReplyToQ* es el nombre local de una cola definida en este gestor de colas.

Si el campo *ReplyToQMgr* está en blanco, el gestor de colas local busca el nombre *ReplyToQ* en sus definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor *ReplyToQMgr* del mensaje transmitido se sustituye por el valor del atributo **RemoteQMgrName** de la definición de la cola remota, y este valor se devuelve en el descriptor de mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, el *ReplyToQMgr* que se transmite con el mensaje es el nombre del gestor de colas local.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para los gestores de colas, o de que el gestor de colas emisor conoce este nombre; esto también es cierto para el nombre transmitido, si el *ReplyToQMgr* se sustituye en el mensaje transmitido.

Si no es necesaria una cola de respuestas, establezca el campo *ReplyToQMgr* en blancos, o (en el lenguaje de programación C) en la serie nula, o en uno o más blancos seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

UserIdentifier (MQCHAR12)

Forma parte del **contexto de identidad** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "MQMD - Descriptor de mensaje" en la página 421 y Contexto de mensaje.

UserIdentifier especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato.

Después de recibir un mensaje, utilice *UserIdentifier* en el campo *AlternateUserId* del parámetro **ObjDesc** de una llamada MQOPEN o MQPUT1 posterior para realizar la comprobación de autorización para el usuario *UserIdentifier* en lugar de la aplicación que realiza la apertura.

Cuando el gestor de colas genera esta información para una llamada MQPUT o MQPUT1 :

- En z/OS, el gestor de colas utiliza *AlternateUserId* del parámetro **ObjDesc** de la llamada MQOPEN o MQPUT1 si se ha especificado la opción MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY. Si no se ha especificado la opción relevante, el gestor de colas utiliza un identificador de usuario determinado desde el entorno.

- En otros entornos, el gestor de colas siempre utiliza un identificador de usuario determinado a partir del entorno.

Cuando el identificador de usuario se determina desde el entorno:

- En z/OS, el gestor de colas utiliza:
  - Para MVS (por lotes), el identificador de usuario de la tarjeta JES JOB o tarea iniciada
  - Para TSO, el identificador de usuario propagado al trabajo durante el envío del trabajo
  - Para CICS, el identificador de usuario asociado a la tarea
  - Para IMS, el identificador de usuario depende del tipo de aplicación:
    - Durante:
      - Regiones BMP no de mensaje
      - Regiones IFP no de mensajes
      - Mensaje BMP y regiones IFP de mensaje que no han emitido una llamada de GU satisfactoria

      el gestor de colas utiliza el identificador de usuario de la tarjeta JES JOB de la región o el identificador de usuario TSO. Si están en blanco o son nulos, utiliza el nombre del bloque de especificación de programa (PSB).
    - Durante:
      - Mensaje BMP y regiones IFP de mensaje que *han* emitido una llamada de GU satisfactoria
      - Regiones MPP

      el gestor de colas utiliza uno de los siguientes:
      - El identificador de usuario conectado asociado con el mensaje
      - El nombre del terminal lógico (LTERM)
      - El identificador de usuario de la tarjeta JES JOB de la región
      - El identificador de usuario TSO
      - El nombre de PSB
- En IBM i, el gestor de colas utiliza el nombre del perfil de usuario asociado con el trabajo de aplicación.
- En UNIX, el gestor de colas utiliza:
  - El nombre de inicio de sesión de la aplicación
  - Identificador de usuario efectivo del proceso si no hay ningún inicio de sesión disponible
  - El identificador de usuario asociado a la transacción, si la aplicación es una transacción CICS
- En sistemas Windows , el gestor de colas utiliza los primeros 12 caracteres del nombre de usuario conectado.

Normalmente, este campo es un campo de salida generado por el gestor de colas, pero para una llamada MQPUT o MQPUT1 puede hacer que este campo sea un campo de entrada/salida y especificar el campo UserIdentification en lugar de dejar que el gestor de colas genere esta información. Especifique MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro Opts PutMsgy especifique un ID de usuario en el campo UserIdentifier si no desea que el gestor de colas genere el campo UserIdentifier para una llamada MQPUT o MQPUT1 .

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el *UserIdentifier* que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de *UserIdentifier* que se mantiene con el mensaje si se conserva (consulte la descripción de

MQPMO_RETAIN para obtener más detalles sobre las publicaciones retenidas) pero no se utiliza como *UserIdentifier* cuando el mensaje se envía como publicación a los suscriptores porque proporcionan un valor para alterar temporalmente *UserIdentifier* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

AccountingToken (MQBYTE32)

Es la señal de contabilidad, parte del *contexto de identidad* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "MQMD - Descriptor de mensaje" en la página 421 ; también consulte Contexto de mensaje.

AccountingToken permite a una aplicación cobrar adecuadamente por el trabajo realizado como resultado del mensaje. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido.

El gestor de colas genera esta información como se indica a continuación:

- El primer byte del campo se establece en la longitud de la información de contabilidad presente en los bytes siguientes; esta longitud está en el rango de cero a 30, y se almacena en el primer byte como un entero binario.
- El segundo y los bytes subsiguientes (según lo especificado por el campo de longitud) se establecen en la información de contabilidad adecuada para el entorno.
  - z/OS En z/OS , la información de contabilidad se establece en:
    - Para el proceso por lotes z/OS , la información de contabilidad de la tarjeta JES JOB o de una sentencia ACCT de JES en la tarjeta EXEC (los separadores de coma se cambian a X'FF '). Esta información se trunca, si es necesario, a 31 bytes.
    - Para TSO, el número de cuenta del usuario.
    - Para CICS, el identificador de unidad de trabajo de LU 6.2 (UEPUOWDS) (26 bytes).
    - Para IMS, el nombre PSB de 8 caracteres concatenado con la señal de recuperación IMS de 16 caracteres.
  - IBM i En IBM i, la información de contabilidad se establece en el código de contabilidad del trabajo.
  - UNIX En UNIX, la información de contabilidad se establece en el identificador de usuario numérico, en caracteres ASCII.
  - Windows En Windows, la información de contabilidad se establece en un identificador de seguridad (SID) de Windows en un formato comprimido. El SID identifica de forma exclusiva el identificador de usuario almacenado en el campo *UserIdentifier* . Cuando el SID se almacena en el campo *AccountingToken* , se omite la autoridad de identificador de 6 bytes (ubicada en el tercer y subsiguiente bytes del SID). Por ejemplo, si el SID de Windows tiene una longitud de 28 bytes, se almacenan 22 bytes de información de SID en el campo *AccountingToken* .
- El último byte (byte 32) del campo de contabilidad se establece en el tipo de señal de contabilidad (en este caso MQACTT_NT_SECURITY_ID, x '0b'):

  MQACTT_CICS_LUOW_ID
  CICS Identificador LUOW.

  Windows **MQACTT_NT_SECURITY_ID**
  Identificador de seguridad de Windows .

  IBM i **MQACTT_OS400_ACCOUNT_TOKEN**
  Señal de contabilidad de IBM i .

UNIX **MQACTT_UNIX_NUMERIC_ID**
Identificador numérico de UNIX .

MQACTT_USER
Señal de contabilidad definida por el usuario.

MQACTT_UNKNOWN
Tipo de señal de contabilidad desconocido.

El tipo de señal de contabilidad se establece en un valor explícito sólo en los entornos siguientes:

– AIX AIX
– IBM i IBM i
– Linux Linux
– Solaris Solaris
– Windows Windows

y para IBM MQ MQI clients conectados a estos sistemas. En otros entornos, el tipo de señal de contabilidad se establece en el valor MQACTT_UNKNOWN. En estos entornos, utilice el campo *PutApplType* para deducir el tipo de señal de contabilidad recibida.

- Todos los demás bytes se establecen en cero binario.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Si no se especifica MQPMO_SET_IDENTITY_CONTEXT ni MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el `AccountingToken` que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de `AccountingToken` que se conserva con el mensaje si se conserva (consulte la descripción de MQPMO_RETAIN en “Opciones de MQPMO (MQLONG)” en la página 508 para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como `AccountingToken` cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente `AccountingToken` en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo es totalmente binario cero.

Es un campo de salida para la llamada MQGET.

Este campo no está sujeto a ninguna conversión basada en el juego de caracteres del gestor de colas; el campo se trata como una serie de bits y no como una serie de caracteres.

El gestor de colas no hace nada con la información de este campo. La aplicación debe interpretar la información si desea utilizarla con fines contables.

Puede utilizar el siguiente valor especial para el campo `AccountingToken` :

MQACT_NONE
No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQACT_NONE_ARRAY también está definida; tiene el mismo valor que MQACT_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial de este campo es MQACT_NONE.

Datos de ApplIdentity(MQCHAR32)

Forma parte del **contexto de identidad** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte “MQMD - Descriptor de mensaje” en la página 421 y Contexto de mensaje.

ApplIdentityData es información definida por la suite de aplicaciones y se puede utilizar para proporcionar información adicional sobre el mensaje o su originador. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Si hay un carácter nulo, el gestor de colas convierte el valor nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_IDENTITY_CONTEXT ni MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el *ApplIdentityData* que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de *ApplIdentityData* que se mantiene con el mensaje si se conserva (consulte la descripción de MQPMO_RETAIN para obtener más detalles sobre las publicaciones retenidas) pero no se utiliza como *ApplIdentityData* cuando el mensaje se envía como publicación a los suscriptores porque proporcionan un valor para alterar temporalmente *ApplIdentityData* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 32 caracteres en blanco en otros lenguajes de programación.

PutApplType (MQLONG)

Es el tipo de aplicación que coloca el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte “MQMD - Descriptor de mensaje” en la página 421 y Contexto de mensaje.

PutApplType puede tener uno de los siguientes tipos estándar. También puede definir sus propios tipos, pero sólo con valores en el rango de MQAT_USER_FIRST a MQAT_USER_LAST.

MQAT_AIX
Aplicación AIX (mismo valor que MQAT_UNIX).

MQAT_AMQP
Aplicación de protocolo AMQP

INTERMEDIARIO
el intermediario de ubicación.

MQAT_CICS
Transacción CICS .

MQAT_CICS_PUENTE
CICS bridge.

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
Aplicación IBM MQ MQI client en PC DOS.

MQAT_DQM
Agente de gestor de colas distribuido.

MQAT_GUARDIÁN
Aplicación Tándem Guardian (mismo valor que MQAT_NSK).

MQAT_IMS
IMS .

MQAT_IMS_BRIDGE
Puente IMS .

MQAT_JAVA
Java.

MQAT_MVS
Aplicación MVS o TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Aplicación de agente.

MQAT_OS390
Aplicación OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
IBM i .

MQAT_QMGR
Gestor de colas.

MQAT_UNIX
UNIX .

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
z/OS .

MQAT_DEFAULT
Tipo de aplicación predeterminado.

Es el tipo de aplicación predeterminado para la plataforma en la que se ejecuta la aplicación.

Nota: El valor de esta constante es específico del entorno. Debido a esto, compile siempre la aplicación utilizando los archivos de cabecera, include o COPY que sean adecuados para la plataforma en la que se ejecutará la aplicación.

MQAT_DESCONOCIDO
Utilice este valor para indicar que el tipo de aplicación es desconocido, aunque haya otra información de contexto.

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

También se puede producir el siguiente valor especial:

MQAT_NO_CONTEXT
Este valor lo establece el gestor de colas cuando se coloca un mensaje sin contexto (es decir, se especifica la opción de contexto MQPMO_NO_CONTEXT).

Cuando se recupera un mensaje, *PutApplType* se puede probar para este valor para decidir si el mensaje tiene contexto (se recomienda que *PutApplType* nunca se establezca en MQAT_NO_CONTEXT, por parte de una aplicación que utilice MQPMO_SET_ALL_CONTEXT, si alguno de los otros campos de contexto no está en blanco).

Cuando el gestor de colas genera esta información como resultado de una colocación de aplicación, el campo se establece en un valor determinado por el entorno. En IBM i, se establece en MQAT_OS400; el gestor de colas nunca utiliza MQAT_CICS en IBM i.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. El valor inicial de este campo es MQAT_NO_CONTEXT.

PutApplNombre (MQCHAR28)

Es el nombre de la aplicación que ha colocado el mensaje y forma parte del *contexto de origen* del mensaje. El contenido difiere entre plataformas y puede diferir entre releases.

Para obtener más información sobre el contexto de mensaje, consulte “MQMD - Descriptor de mensaje” en la página 421 y Contexto de mensaje.

V 9.1.2 A partir de IBM MQ 9.1.2 , puede especificar el nombre de aplicación en lenguajes de programación adicionales. Para obtener más información, consulte Especificar el nombre de la aplicación en lenguajes de programación soportados.

El formato de *PutApplName* depende del valor de *PutApplType* y puede cambiar de un release a otro. Los cambios son raros, pero ocurren si el entorno cambia.

Cuando el gestor de colas establece este campo (es decir, para todas las opciones excepto MQPMO_SET_ALL_CONTEXT), establece el campo en un valor determinado por el entorno:

- z/OS En z/OS, el gestor de colas utiliza:
  - Para el proceso por lotes z/OS , el nombre de trabajo de 8 caracteres de la tarjeta JES JOB
  - Para TSO, el identificador de usuario TSO de 7 caracteres
  - Para CICS, el ID de aplicación de 8 caracteres, seguido del ID de aplicación de 4 caracteres
  - Para IMS, el identificador del sistema IMS de 8 caracteres, seguido del nombre PSB de 8 caracteres
  - Para XCF, el nombre de grupo XCF de 8 caracteres, seguido del nombre de miembro XCF de 16 caracteres
  - Para un mensaje generado por un gestor de colas, los primeros 28 caracteres del nombre del gestor de colas
  - Para colas distribuidas sin CICS, el nombre de trabajo de 8 caracteres del iniciador de canal seguido del nombre de 8 caracteres del módulo que se coloca en la cola de mensajes no entregados seguido de un identificador de tarea de 8 caracteres.

  El nombre o los nombres se rellenan cada uno a la derecha con espacios en blanco, al igual que cualquier espacio en el resto del campo. Cuando hay más de un nombre, no hay ningún separador entre ellos.
- Windows En sistemas Windows , el gestor de colas utiliza los nombres siguientes:
  - Para una aplicación CICS , el nombre de transacción CICS
  - Para una aplicación que no es deCICS , los 28 caracteres más a la derecha del nombre completo del ejecutable
- IBM i En IBM i, el gestor de colas utiliza el nombre de trabajo completo.
- UNIX En UNIX, el gestor de colas utiliza los nombres siguientes:
  - Para una aplicación CICS , el nombre de transacción CICS
  - Para una aplicación que no es deCICS , MQ solicita al sistema operativo el nombre del proceso. Se devuelve como nombre de archivo de programa, sin vía de acceso completa. A continuación, MQ coloca este nombre de proceso en el MQMD de MQMD.PutApplName como se indica a continuación:

AIX

Si el nombre es menor o igual que 28 bytes, se inserta el nombre, rellenado a la derecha con espacios.

Si el nombre es mayor que 28 bytes, se insertan los 28 bytes más a la izquierda del nombre.

Linux y Solaris

Si el nombre es menor o igual que 15 bytes, se inserta el nombre, rellenado a la derecha con espacios.

Si el nombre es mayor que 15 bytes, se insertan los 15 bytes más a la izquierda del nombre, rellenados a la derecha con espacios.

Por ejemplo, si ejecuta `/opt/mqm/samp/bin/amqsput QNAME QMNAME`, el nombre de PutApples `'amqsput'`. Hay 21 caracteres de espacio de relleno en este campo MQCHAR28 . Tenga en cuenta que la vía de acceso completa que incluye `/opt/mqm/samp/bin` no se incluye en el nombre de PutAppl.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

PutDate (MQCHAR8)

Es la fecha en la que se ha colocado el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "MQMD - Descriptor de mensaje" en la página 421 y Contexto de mensaje.

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- AAAAMMDD

donde los caracteres representan:

AAAA
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, la fecha es la fecha en la que se ha colocado el mensaje y no la fecha en la que se ha confirmado la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_PUT_DATE_LENGTH. El valor inicial de este campo es la serie nula en C y 8 caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)

Es la hora a la que se ha colocado el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "MQMD - Descriptor de mensaje" en la página 421 y Contexto de mensaje.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan (en orden):

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte la nota)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, es posible que en raras ocasiones se devuelvan 60 o 61 durante los segundos en *PutTime*. Esto sucede cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, el tiempo es el momento en que se colocó el mensaje, y no el momento en que se comprometió la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la serie nula en C y 8 caracteres en blanco en otros lenguajes de programación.

Datos de ApplOrigin(MQCHAR4)

Forma parte del *contexto de origen* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "MQMD - Descriptor de mensaje" en la página 421 y Contexto de mensaje.

`ApplOriginData` es información definida por la suite de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza.

El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_APPL_ORIGIN_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 4 caracteres en blanco en otros lenguajes de programación.

Cuando se publica el mensaje, aunque `ApplOriginData` está establecido, está en blanco en la suscripción que recibe.

GroupId (MQBYTE24)

Es una serie de bytes que se utiliza para identificar el grupo de mensajes o el mensaje lógico al que pertenece el mensaje físico. *`GroupId`* también se utiliza si se permite la segmentación para el mensaje. En todos estos casos, *`GroupId`* tiene un valor no nulo y uno o varios de los distintivos siguientes se establecen en el campo *`MsgFlags`* :

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

Si no se establece ninguno de estos distintivos, *`GroupId`* tiene el valor nulo especial MQGI_NONE.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se ha especificado MQMO_MATCH_GROUP_ID.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que *`GroupId`* esté establecido en un valor adecuado.

Los grupos de mensajes y segmentos sólo se pueden procesar correctamente si el identificador de grupo es exclusivo. Por este motivo, las aplicaciones *no deben generar sus propios identificadores de grupo* ; en su lugar, las aplicaciones deben realizar una de las acciones siguientes:

- Si se especifica MQPMO_LOGICAL_ORDER, el gestor de colas genera automáticamente un identificador de grupo exclusivo para el primer mensaje del grupo o segmento del mensaje lógico, y utiliza dicho identificador de grupo para los mensajes restantes del grupo o segmentos del mensaje lógico, por lo que la aplicación no necesita realizar ninguna acción especial. Este es el procedimiento recomendado.
- Si no se especifica MQPMO_LOGICAL_ORDER, la aplicación debe solicitar al gestor de colas que genere el identificador de grupo, estableciendo *`GroupId`* en MQGI_NONE en la primera llamada MQPUT o MQPUT1 para un mensaje del grupo o segmento del mensaje lógico. El identificador de grupo devuelto por el gestor de colas en la salida de esa llamada debe utilizarse entonces para los mensajes restantes en el grupo o segmentos del mensaje lógico. Si un grupo de mensajes contiene mensajes segmentados, se debe utilizar el mismo identificador de grupo para todos los segmentos y mensajes del grupo.

  Cuando no se especifica MQPMO_LOGICAL_ORDER, los mensajes de grupos y segmentos de mensajes lógicos se pueden poner en cualquier orden (por ejemplo, en orden inverso), pero el identificador de grupo debe ser asignado por la llamada *first* MQPUT o MQPUT1 que se emite para cualquiera de estos mensajes.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje si el objeto abierto es una sola cola y no una lista de distribución, pero lo deja sin modificar si el objeto abierto es una lista de distribución. En este último caso, si la aplicación necesita conocer los identificadores de grupo generados, la aplicación debe proporcionar registros MQPMR que contengan el campo *`GroupId`* .

En la entrada a la llamada MQGET, el gestor de colas utiliza el valor descrito en Tabla 495 en la página 392. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

Se define el siguiente valor especial:

MQGI_NONE
No se ha especificado ningún identificador de grupo.

El valor es cero binario para la longitud del campo. Es el valor que se utiliza para los mensajes que no están en grupos, no en segmentos de mensajes lógicos y para los que no se permite la segmentación.

Para el lenguaje de programación C, la constante MQGI_NONE_ARRAY también está definida; tiene el mismo valor que MQGI_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_GROUP_ID_LENGTH. El valor inicial de este campo es MQGI_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgSeqNumber (MQLONG)

Es el número de secuencia de un mensaje lógico dentro de un grupo.

Los números de secuencia empiezan en 1 y aumentan en 1 para cada nuevo mensaje lógico del grupo, hasta un máximo de 999 999 999. Un mensaje físico que no está en un grupo tiene un número de secuencia de 1.

La aplicación no tiene que establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se ha especificado MQMO_MATCH_MSG_SEQ_NUMBER.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que *MsgSeqNumber* esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

En la entrada de la llamada MQGET, el gestor de colas utiliza el valor que se muestra en Tabla 495 en la página 392. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es uno. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

Offset (MQLONG)

Es el desplazamiento en bytes de los datos del mensaje físico desde el inicio del mensaje lógico del que forman parte los datos. Estos datos se denominan *segmento*. El desplazamiento está en el rango de 0 a 999.999.999. Un mensaje físico que no es un segmento de un mensaje lógico tiene un desplazamiento de cero.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se ha especificado MQMO_MATCH_OFFSET.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación no cumple estas condiciones, o la llamada es MQPUT1, la aplicación debe asegurarse de que *Offset* esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

Para un mensaje de informe que informa sobre un segmento de un mensaje lógico, el campo *OriginalLength* (siempre que no sea MQOL_UNDEFINED) se utiliza para actualizar el desplazamiento en la información de segmento retenida por el gestor de colas.

En la entrada de la llamada MQGET, el gestor de colas utiliza el valor que se muestra en Tabla 495 en la página 392. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es cero. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgFlags (MQLONG)

Los MsgFlags son distintivos que especifican atributos del mensaje o controlan su proceso.

Los MsgFlags se dividen en las categorías siguientes:

- Distintivos de segmentación
- Distintivos de estado

Distintivos de segmentación: cuando un mensaje es demasiado grande para una cola, normalmente falla un intento de colocar el mensaje en la cola. La segmentación es una técnica mediante la cual el gestor de colas o la aplicación divide el mensaje en partes más pequeñas denominadas segmentos, y coloca cada segmento en la cola como un mensaje físico independiente. La aplicación que recupera el mensaje puede recuperar los segmentos uno por uno o solicitar al gestor de colas que vuelva a ensamblar los segmentos en un único mensaje devuelto por la llamada MQGET. Esto último se consigue especificando la opción MQGMO_COMPLETE_MSG en la llamada MQGET y proporcionando un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo. (Consulte "MQGMO-Opciones de obtención de mensajes" en la página 367 para obtener detalles de la opción MQGMO_COMPLETE_MSG.) Un mensaje se puede segmentar en el gestor de colas emisor, en un gestor de colas intermedio o en el gestor de colas de destino.

Puede especificar una de las siguientes opciones para controlar la segmentación de un mensaje:

MQMF_SEGMENTATION_INHIBIDO
Esta opción impide que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción impide que el segmento se divida en segmentos más pequeños.

El valor de este distintivo es cero binario. Éste es el valor predeterminado.

MQMF_SEGMENTATION_ALLOWED
Esta opción permite que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción permite que el segmento se divida en segmentos más pequeños. MQMF_SEGMENTATION_ALLOWED se puede establecer sin establecer MQMF_SEGMENT o MQMF_LAST_SEGMENT.

- En z/OS, el gestor de colas no da soporte a la segmentación de mensajes. Si un mensaje es demasiado grande para la cola, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_MSG_TOO_BIG_FOR_Q. Sin embargo, todavía se puede especificar la opción MQMF_SEGMENTATION_ALLOWED y permite segmentar el mensaje en un gestor de colas remoto.

Cuando el gestor de colas segmenta un mensaje, el gestor de colas activa el distintivo MQMF_SEGMENT en la copia del MQMD que se envía con cada segmento, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 . Para el último segmento del mensaje lógico, el gestor de colas también activa el distintivo MQMF_LAST_SEGMENT en el MQMD que se envía con el segmento.

Nota: Tenga cuidado al colocar mensajes con MQMF_SEGMENTATION_ALLOWED pero sin MQPMO_LOGICAL_ORDER. Si el mensaje es:

- No es un segmento, y
- No en un grupo, y
- No se reenvía,

la aplicación debe restablecer el campo *GroupId* en MQGI_NONE antes de *cada llamada* MQPUT o MQPUT1 , para que el gestor de colas pueda generar un identificador de grupo exclusivo para cada

mensaje. Si esto no se hace, los mensajes no relacionados pueden tener el mismo identificador de grupo, lo que puede llevar a un proceso incorrecto posteriormente. Consulte las descripciones del campo *GroupId* y la opción MQPMO_LOGICAL_ORDER para obtener más información sobre cuándo restablecer el campo *GroupId* .

El gestor de colas divide los mensajes en segmentos según sea necesario para que los segmentos (más los datos de cabecera necesarios) quepan en la cola. Sin embargo, existe un límite inferior para el tamaño de un segmento generado por el gestor de colas, y sólo el último segmento creado a partir de un mensaje puede ser menor que este límite (el límite inferior para el tamaño de un segmento generado por la aplicación es de un byte). Los segmentos generados por el gestor de colas pueden tener una longitud desigual. El gestor de colas procesa el mensaje como se indica a continuación:

- Los formatos definidos por el usuario se dividen en límites que son múltiplos de 16 bytes; el gestor de colas no genera segmentos que sean menores de 16 bytes (que no sean el último segmento).
- Los formatos incorporados distintos de MQFMT_STRING se dividen en puntos adecuados a la naturaleza de los datos presentes. Sin embargo, el gestor de colas nunca divide un mensaje en medio de una estructura de cabecera IBM MQ . Esto significa que el gestor de colas no puede dividir más un segmento que contenga una única estructura de cabecera MQ y, como resultado, el tamaño de segmento mínimo posible para ese mensaje es mayor que 16 bytes.

  El segundo segmento o segmento posterior generado por el gestor de colas empieza por uno de los siguientes:

  – Una estructura de cabecera MQ
  – El inicio de los datos del mensaje de aplicación
  – Parte del camino a través de los datos del mensaje de aplicación
- MQFMT_STRING se divide sin tener en cuenta la naturaleza de los datos presentes (SBCS, DBCS o SBCS/DBCS mixto). Cuando la serie es DBCS o SBCS/DBCS mixto, esto puede dar como resultado segmentos que no se pueden convertir de un juego de caracteres a otro. El gestor de colas nunca divide los mensajes MQFMT_STRING en segmentos menores de 16 bytes (que no sean el último segmento).
- El gestor de colas establece los campos *Format*, *CodedCharSetId*y *Encoding* en el MQMD de cada segmento para describir correctamente los datos presentes en el *inicio* del segmento; el nombre de formato es el nombre de un formato incorporado o el nombre de un formato definido por el usuario.
- Se modifica el campo *Report* en el MQMD de segmentos con *Offset* mayor que cero. Para cada tipo de informe, si la opción de informe es MQRO_ * _WITH_DATA, pero el segmento no puede contener ninguno de los primeros 100 bytes de datos de usuario (es decir, los datos que siguen a las estructuras de cabecera de IBM MQ que pueden estar presentes), la opción de informe se cambia a MQRO_ *.

El gestor de colas sigue las reglas anteriores, pero de lo contrario divide los mensajes de forma imprevisible; no realice suposiciones sobre dónde se divide un mensaje.

Para los mensajes *persistentes* , el gestor de colas sólo puede realizar la segmentación dentro de una unidad de trabajo:

- Si la llamada MQPUT o MQPUT1 está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de segmentación, el gestor de colas elimina los segmentos que se colocaron en la cola como resultado de la llamada anómala. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo sólo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente. Si la llamada falla, el gestor de colas restituye la unidad de trabajo.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar la segmentación. Si el mensaje no requiere segmentación, la llamada puede seguir siendo

satisfactoria. Pero si el mensaje requiere segmentación, la llamada falla con el código de razón MQRC_UOW_NOT_AVAILABLE.

Para los mensajes *no persistentes* , el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar la segmentación.

Tenga especial cuidado al convertir datos en mensajes que se pueden segmentar:

- Si la aplicación receptora convierte datos en la llamada MQGET y especifica la opción MQGMO_COMPLETE_MSG, a la salida de conversión de datos se le pasa el mensaje completo para que la salida lo convierta y el hecho de que el mensaje se haya segmentado es evidente para la salida.
- Si la aplicación receptora recupera un segmento a la vez, se invoca la salida de conversión de datos para convertir un segmento a la vez. Por lo tanto, la salida debe convertir los datos de un segmento independientemente de los datos de cualquiera de los otros segmentos.

  Si la naturaleza de los datos del mensaje es tal que la segmentación arbitraria de los datos en límites de 16 bytes puede dar como resultado segmentos que la salida no puede convertir, o el formato es MQFMT_STRING y el juego de caracteres es DBCS o SBCS/DBCS mixto, la aplicación emisora debe crear y colocar los segmentos, especificando MQMF_SEGMENTATION_INHIBIDO para suprimir la segmentación adicional. De esta forma, la aplicación emisora puede asegurarse de que cada segmento contiene información suficiente para permitir que la salida de conversión de datos convierta el segmento correctamente.
- Si se especifica la conversión del emisor para un agente de canal de mensajes (MCA) de envío, el MCA sólo convierte los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir los mensajes que son segmentos.

Este distintivo es un distintivo de entrada en las llamadas MQPUT y MQPUT1 y un distintivo de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite el valor del distintivo en el campo *Segmentation* en MQGMO.

El valor inicial de este distintivo es MQMF_SEGMENTATION_INHIBIDO.

Distintivos de estado: son distintivos que indican si el mensaje físico pertenece a un grupo de mensajes, es un segmento de un mensaje lógico, ambos o ninguno. Se pueden especificar uno o más de los siguientes valores en la llamada MQPUT o MQPUT1 , o la llamada MQGET los puede devolver:

MQMF_MSG_IN_GROUP
El mensaje es miembro de un grupo.

MQMF_LAST_MSG_IN_GROUP
El mensaje es el último mensaje lógico de un grupo.

Si se establece este distintivo, el gestor de colas activa MQMF_MSG_IN_GROUP en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Es válido que un grupo conste de un solo mensaje lógico. Si este es el caso, se establece MQMF_LAST_MSG_IN_GROUP, pero el campo *MsgSeqNumber* tiene el valor uno.

SEGMENTO MQMF_SEGMENT
El mensaje es un segmento de un mensaje lógico.

Cuando se especifica MQMF_SEGMENT sin MQMF_LAST_SEGMENT, la longitud de los datos de mensaje de aplicación en el segmento (*excluyendo* las longitudes de las estructuras de cabecera IBM MQ que pueden estar presentes) debe ser como mínimo una. Si la longitud es cero, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_SEGMENT_LENGTH_ZERO.

En z/OS, esta opción no está soportada si el mensaje se está colocando en una cola que tiene un tipo de índice de MQIT_GROUP_ID.

MQMF_LAST_SEGMENT
El mensaje es el último segmento de un mensaje lógico.

Si se establece este distintivo, el gestor de colas activa MQMF_SEGMENT en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Un mensaje lógico sólo puede constar de un segmento. Si es así, se establece MQMF_LAST_SEGMENT, pero el campo *Offset* tiene el valor cero.

Cuando se especifica MQMF_LAST_SEGMENT, la longitud de los datos del mensaje de aplicación en el segmento (*excluyendo* las longitudes de las estructuras de cabecera que pueden estar presentes) puede ser cero.

En z/OS, esta opción no está soportada si el mensaje se está colocando en una cola que tiene un tipo de índice de MQIT_GROUP_ID.

La aplicación debe asegurarse de que estos distintivos se establecen correctamente al transferir mensajes. Si se especifica MQPMO_LOGICAL_ORDER, o se ha especificado en la llamada MQPUT anterior para el descriptor de contexto de cola, los valores de los distintivos deben ser coherentes con la información de grupo y segmento retenida por el gestor de colas para el descriptor de contexto de cola. Las condiciones siguientes se aplican a las llamadas MQPUT *sucesivas* para el descriptor de contexto de cola cuando se especifica MQPMO_LOGICAL_ORDER:

- Si no hay ningún grupo o mensaje lógico actual, todos estos distintivos (y combinaciones de ellos) son válidos.
- Una vez que se ha especificado MQMF_MSG_IN_GROUP, debe permanecer activado hasta que se especifique MQMF_LAST_MSG_IN_GROUP. La llamada falla con el código de razón MQRC_INCOMPLETE_GROUP si no se cumple esta condición.
- Una vez que se ha especificado MQMF_SEGMENT, debe permanecer activado hasta que se especifique MQMF_LAST_SEGMENT. La llamada falla con el código de razón MQRC_INCOMPLETE_MSG si no se cumple esta condición.
- Una vez que se ha especificado MQMF_SEGMENT sin MQMF_MSG_IN_GROUP, MQMF_MSG_IN_GROUP debe permanecer *desactivado* hasta que se haya especificado MQMF_LAST_SEGMENT. La llamada falla con el código de razón MQRC_INCOMPLETE_MSG si no se cumple esta condición.

Orden físico en una cola muestra las combinaciones válidas de los distintivos y los valores utilizados para diversos campos.

Estos distintivos son distintivos de entrada en las llamadas MQPUT y MQPUT1 y distintivos de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite los valores de los distintivos en los campos *GroupStatus* y *SegmentStatus* en MQGMO.

No puede utilizar mensajes agrupados o segmentados con la publicación/suscripción.

Distintivos predeterminados: se puede especificar lo siguiente para indicar que el mensaje tiene atributos predeterminados:

MQMF_NONE
Sin distintivos de mensaje (atributos de mensaje predeterminados).

Esto inhibe la segmentación e indica que el mensaje no está en un grupo y no es un segmento de un mensaje lógico. MQMF_NONE está definido para ayudar a la documentación del programa. No se pretende que este distintivo se utilice con ningún otro, pero como su valor es cero, no se puede detectar dicho uso.

El campo *MsgFlags* se particiona en subcampos; para obtener detalles, consulte “Opciones de informe y distintivos de mensaje” en la página 919.

El valor inicial de este campo es MQMF_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

OriginalLength (MQLONG)

Este campo sólo es relevante para los mensajes de informe que son segmentos. Especifica la longitud del segmento de mensaje con el que se relaciona el mensaje de informe; no especifica la longitud del mensaje lógico del que forma parte el segmento o la longitud de los datos del mensaje de informe.

Nota: Al generar un mensaje de informe para un mensaje que es un segmento, el gestor de colas y el agente de canal de mensajes copian en el MQMD del mensaje de informe los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags*, del mensaje original. Como resultado, el mensaje de informe también es un segmento. Las aplicaciones que generan mensajes de informe deben hacer lo mismo y establecer el campo *OriginalLength* correctamente.

Se define el siguiente valor especial:

MQOL_UNDEFINED
No se ha definido la longitud original del mensaje.

OriginalLength es un campo de entrada en las llamadas MQPUT y MQPUT1 , pero el valor que proporciona la aplicación sólo se acepta en determinadas circunstancias:

- Si el mensaje que se está colocando es un segmento y también es un mensaje de informe, el gestor de colas acepta el valor especificado. El valor debe ser:
  - Mayor que cero si el segmento no es el último segmento
  - No menor que cero si el segmento es el último segmento
  - No menor que la longitud de los datos presentes en el mensaje

  Si no se cumplen estas condiciones, la llamada falla con el código de razón MQRC_ORIGINAL_LENGTH_ERROR.
- Si el mensaje que se está colocando es un segmento pero no un mensaje de informe, el gestor de colas ignora el campo y utiliza en su lugar la longitud de los datos del mensaje de aplicación.
- En todos los demás casos, el gestor de colas ignora el campo y utiliza el valor MQOL_UNDEFINED en su lugar.

Es un campo de salida en la llamada MQGET.

El valor inicial de este campo es MQOL_UNDEFINED. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MQMDE-Extensión de descriptor de mensaje

La estructura MQMDE describe los datos que a veces se producen antes de los datos del mensaje de aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2 , pero no en el MQMD version-1 .

Disponibilidad

Todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

MQFMT_MD_EXTENSION

Juego de caracteres y codificación

Los datos de MQMDE deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE para el lenguaje de programación C.

Establezca el juego de caracteres y la codificación de MQMDE en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQMDE está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQMDE (todos los demás casos).

Si el MQMDE no está en el juego de caracteres y la codificación del gestor de colas, el MQMDE se acepta pero no se respeta, es decir, el MQMDE se trata como datos de mensaje.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas. Aunque los campos numéricos de la estructura MQMD en las llamadas MQPUT, MQPUT1y MQGET deben estar en la codificación Micro Focus COBOL, los campos numéricos de la estructura MQMDE deben estar en la codificación del gestor de colas. Esto último lo proporciona MQENC_NATIVE para el lenguaje de programación C, y tiene el valor 546.

Utilización

Las aplicaciones que utilizan un MQMD version-2 no encontrarán una estructura MQMDE. Sin embargo, las aplicaciones especializadas y las aplicaciones que siguen utilizando un MQMD de version-1 , pueden encontrar un MQMDE en algunas situaciones. La estructura MQMDE puede producirse en las circunstancias siguientes:

- Especificado en las llamadas MQPUT y MQPUT1
- Devuelto por la llamada MQGET
- En mensajes en colas de transmisión

MQMDE especificado en llamadas MQPUT y MQPUT1

En las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD version-1 , la aplicación puede opcionalmente añadir un prefijo a los datos de mensaje con un MQMDE, estableciendo el campo *Format* de MQMD en MQFMT_MD_EXTENSION para indicar que existe un MQMDE. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE. Los valores predeterminados que utiliza el gestor de colas son los mismos que los valores iniciales para la estructura; consulte Tabla 503 en la página 476.

Si la aplicación proporciona un version-2 MQMD *y* prefija los datos de mensaje de aplicación con un MQMDE, las estructuras se procesan tal como se muestra en la tabla siguiente.

Tabla 502. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o MQPUT1 para MQMDE

MQMD Version	Valores de los campos version-2	Valores de los campos correspondientes en MQMDE	Acción realizada por el gestor de colas
1	-	Válido	MQMDE se respeta
2	Valor predeterminado	Válido	MQMDE se respeta
2	No predeterminado	Válido	MQMDE se trata como datos de mensaje
1 o 2	Cualquiera	No válido	La llamada falla con un código de razón adecuado
1 o 2	Cualquiera	MQMDE está en el juego de caracteres o codificación incorrectos, o es una versión no soportada	MQMDE se trata como datos de mensaje

Nota: En z/OS, si la aplicación especifica un MQMD version-1 con un MQMDE, el gestor de colas valida el MQMDE sólo si la cola tiene un *IndexType* de MQIT_GROUP_ID.

Hay un caso especial. Si la aplicación utiliza un MQMD version-2 para colocar un mensaje que es un segmento (es decir, se establece el distintivo MQMF_SEGMENT o MQMF_LAST_SEGMENT), y el nombre de formato del MQMD es MQFMT_DEAD_LETTER_HEADER, el gestor de colas genera una estructura MQMDE

y la inserta *entre* la estructura MQDLH y los datos que le siguen. En el MQMD que el gestor de colas conserva con el mensaje, los campos version-2 se establecen en sus valores predeterminados.

Varios de los campos que existen en MQMD version-2 pero no en MQMD version-1 son campos de entrada/salida en MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .

MQMDE devuelto por la llamada MQGET

En la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. El gestor de colas establece el campo *Format* en MQMD en el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Si la aplicación proporciona un MQMDE al inicio del parámetro **Buffer** , el MQMDE se ignora. Al volver de la llamada MQGET, se sustituye por el MQMDE para el mensaje (si es necesario), o se sobrescribe por los datos del mensaje de aplicación (si no es necesario el MQMDE).

Si la llamada MQGET devuelve un MQMDE, los datos del MQMDE suelen estar en el juego de caracteres y la codificación del gestor de colas. Sin embargo, MQMDE puede estar en algún otro juego de caracteres y codificación si:

- MQMDE se ha tratado como datos en la llamada MQPUT o MQPUT1 (consulte Tabla 502 en la página 475 para conocer las circunstancias que pueden causar esto).
- El mensaje se ha recibido de un gestor de colas remoto conectado mediante una conexión TCP y el agente de canal de mensajes (MCA) receptor no se ha configurado correctamente.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas (consulte más arriba).

MQMDE en mensajes en colas de transmisión

Los mensajes de las colas de transmisión tienen como prefijo la estructura MQXQH, que contiene un MQMD version-1 . Un MQMDE también puede estar presente, situado entre la estructura MQXQH y los datos del mensaje de aplicación, pero normalmente sólo está presente si uno o varios de los campos de MQMDE tienen un valor no predeterminado.

También se pueden producir otras estructuras de cabecera MQ entre la estructura MQXQH y los datos del mensaje de aplicación. Por ejemplo, cuando la cabecera de mensaje no entregado MQDLH está presente y el mensaje no es un segmento, el orden es:

- MQXQH (que contiene un MQMD version-1)
- MQMDE
- MQDLH
- datos de mensaje de aplicación

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 503. Campos en MQMDE para MQMDE

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQMDE_STRUC_ID	'MDE¬'
Versión (número de versión de estructura)	MQMDE_VERSION_2	2

Tabla 503. Campos en MQMDE para MQMDE (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
StrucLength (longitud de la estructura MQMDE)	MQMDE_LENGTH_2	72
Codificación (codificación numérica de datos que sigue a MQMDE)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (identificador de juego de caracteres de los datos que siguen a MQMDE)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de los datos que siguen a MQMDE)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos generales)	MQMDEF_NONE	0
GroupId (identificador de grupo)	MQGI_NONE	Nulos
MsgSeqNúmero (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
Desplazamiento (desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0
MsgFlags (distintivos de mensaje)	MQMF_NONE	0
OriginalLength (longitud del mensaje original)	MQOL_UNDEFINED	-1

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQMDE_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

El valor debe ser:

MQMDE_STRUC_ID
Identificador de la estructura de extensión del descriptor de mensaje.

Para el lenguaje de programación C, también se define la constante MQMDE_STRUC_ID_ARRAY; tiene el mismo valor que MQMDE_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQMDE_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQMDE_VERSION_2
Estructura de extensión del descriptor de mensaje Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

MQMDE_CURRENT_VERSION
Versión actual de la estructura de extensión del descriptor de mensaje.

El valor inicial de este campo es MQMDE_VERSION_2.

StrucLength (MQLONG)

Es la longitud de la estructura MQMDE; se define el valor siguiente:

MQMDE_LENGTH_2
Longitud de la estructura de extensión del descriptor de mensaje version-2 .

El valor inicial de este campo es MQMDE_LENGTH_2.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a la estructura MQMDE; no se aplica a los datos numéricos de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Consulte el campo *Encoding* descrito en "MQMD - Descriptor de mensaje" en la página 421 para obtener más información sobre las codificaciones de datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQMDE; no se aplica a los datos de tipo carácter de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a la estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Consulte el campo *Format* descrito en"MQMD - Descriptor de mensaje" en la página 421para obtener más información sobre los nombres de formato.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Se puede especificar el distintivo siguiente:

MQMDEF_NONE
Sin distintivos.

El valor inicial de este campo es MQMDEF_NONE.

GroupId (MQBYTE24)

Consulte el campo *GroupId* descrito en"MQMD - Descriptor de mensaje" en la página 421. El valor inicial de este campo es MQGI_NONE.

MsgSeqNumber (MQLONG)

Consulte el campo *MsgSeqNumber* descrito en"MQMD - Descriptor de mensaje" en la página 421. El valor inicial de este campo es 1.

Offset (MQLONG)

Consulte el campo *Offset* descrito en“MQMD - Descriptor de mensaje” en la página 421. El valor inicial de este campo es 0.

MsgFlags (MQLONG)

Consulte el campo *MsgFlags* descrito en “MQMD - Descriptor de mensaje” en la página 421. El valor inicial de este campo es MQMF_NONE.

OriginalLength (MQLONG)

Consulte el campo *OriginalLength* descrito en“MQMD - Descriptor de mensaje” en la página 421. El valor inicial de este campo es MQOL_UNDEFINED.

MQMHBO-Descriptor de mensaje para opciones de almacenamiento intermedio

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los almacenamientos intermedios a partir de los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUF.

Juego de caracteres y codificación

Los datos de MQMHBO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 504. Campos en MQMHBO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQMHBO_STRUC_ID	'MHBO'
Versión (número de versión de estructura)	MQMHBO_VERSION_1	1
Options (opciones que controlan la acción de MQMHBUF)	MQMHBO_PROPERTIES_IN_MQRFH2	

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQMHBO_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Estructura de descriptor de contexto de mensaje para opciones de almacenamiento intermedio-campo StrucId

Es el identificador de estructura. El valor debe ser:

MQMHBO_STRUC_ID
: Identificador del descriptor de mensaje para la estructura de opciones de almacenamiento intermedio.

  Para el lenguaje de programación C, también se define la constante MQMHBO_STRUC_ID_ARRAY; tiene el mismo valor que MQMHBO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_STRUC_ID.

Versión (MQLONG)

Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQMHBO_VERSION_1
: Número de versión del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

La constante siguiente especifica el número de versión de la versión actual:

MQMHBO_CURRENT_VERSION
: Versión actual del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_VERSION_1.

Opciones (MQLONG)

Descriptor de contexto de mensaje para estructura de opciones de almacenamiento intermedio-Campo Opciones

Estas opciones controlan la acción de MQMHBUF.

Debe especificar la opción siguiente:

MQMHBO_PROPERTIES_IN_MQRFH2
: Al convertir propiedades de un descriptor de mensaje en un almacenamiento intermedio, conviértalas al formato MQRFH2 .

Opcionalmente, también puede especificar la opción siguiente. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

MQMHBO_DELETE_PROPERTIES
: Las propiedades que se añaden al almacenamiento intermedio se suprimen del descriptor de contexto de mensaje. Si la llamada falla, no se suprimen las propiedades.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_PROPERTIES_IN_MQRFH2.

MQOD-Descriptor de objeto

La estructura MQOD se utiliza para especificar un objeto por nombre. La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1 .
Son válidos los siguientes tipos de objeto:

- Cola o lista de distribución
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Disponibilidad

Todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a dichos sistemas.

Versión

La versión actual de MQOD es MQOD_VERSION_4. Las aplicaciones que desee portar entre varios entornos deben asegurarse de que la versión necesaria de MQOD esté soportada en todos los entornos afectados. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQOD soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQOD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Para abrir una lista de distribución, *Version* debe ser MQOD_VERSION_2 o superior.

Juego de caracteres y codificación

Los datos de MQOD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQOD_ID_STRUCD	'OD¬¬'
Versión (número de versión de estructura)	MQOD_VERSION_1	1
ObjectType (tipo de objeto)	MQOT_Q	1
ObjectName (nombre de objeto)	Ninguna	Serie nula o espacios en blanco
ObjectQMgrNombre (nombre de gestor de colas de objeto)	Ninguna	Serie nula o espacios en blanco
DynamicQName (nombre de cola dinámica)	Ninguna	'CSQ.*' en z/OS ; 'AMQ.*' de lo contrario
AlternateUserId (identificador de usuario alternativo)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_2.		
RecsPresent (número de registros de objeto presentes)	Ninguna	0
KnownDestKnownDest (número de colas locales abiertas correctamente)	Ninguna	0
UnknownDestCount (número de colas remotas abiertas correctamente)	Ninguna	0
InvalidDestCount (número de colas que no se han podido abrir)	Ninguna	0
Desplazamiento deObjectRec (desplazamiento del primer registro de objeto desde el inicio de MQOD)	Ninguna	0
ResponseRecDesplazamiento (desplazamiento del primer registro de respuesta desde el inicio de MQOD)	Ninguna	0
ObjectRecPtr (dirección del primer registro de objeto)	Ninguna	Puntero nulo o bytes nulos
ResponseRecPtr (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_3.		
AlternateSecurityId (identificador de seguridad alternativo)	MQSID_NONE	Nulos
ResolvedQName (nombre de cola resuelto)	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrNombre (nombre de gestor de colas resuelto)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_4.		

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
ObjectString (nombre de objeto largo)	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
SelectionString (serie de selección)	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
ResObjectString (nombre de objeto largo resuelto)	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
ResolvedType (tipo de objeto resuelto)	MQOT_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQOD_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQOD MyOD = {MQOD_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura; el valor debe ser:

MQOD_ID_STRUCD
Identificador de la estructura de descriptor de objeto.

Para el lenguaje de programación C, también se define la constante MQOD_STRUC_ID_ARRAY; tiene el mismo valor que MQOD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQOD_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura; el valor debe ser uno de los siguientes:

MQOD_VERSION_1
Estructura del descriptor de objeto Version-1 .

MQOD_VERSION_2
Estructura del descriptor de objeto Version-2 .

MQOD_VERSION_3
Estructura del descriptor de objeto Version-3 .

MQOD_VERSION_4
Estructura del descriptor de objeto Version-4 .

Todas las versiones están soportadas en todos los entornos IBM MQ V7.0 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQOD_CURRENT_VERSION
Versión actual de la estructura de descriptor de objeto.

Siempre es un campo de entrada. El valor inicial de este campo es MQOD_VERSION_1.

ObjectType (MQLONG)

El tipo de objeto que se está nombrando en el descriptor de objeto. Los valores posibles son:

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente. El nombre del objeto se encuentra en el campo *ObjectName* .

MQOT_Q
Cola. El nombre del objeto se encuentra en el campo *ObjectName* .

MQOT_NAMELIST
Lista de nombres. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_PROCESS
. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_Q_MGR
Gestor de colas. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_TOPIC
. El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*.

Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

Siempre es un campo de entrada. El valor inicial de este campo es MQOT_Q.

ObjectName (MQCHAR48)

Es el nombre local del objeto tal como se define en el gestor de colas identificado por *ObjectQMgrName*. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el nulo y los caracteres que le siguen se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS:

  - Evite los nombres que empiezan o terminan con un guión bajo; no pueden ser procesados por las operaciones y los paneles de control.
  - El carácter de porcentaje tiene un significado especial para RACF. Si se utiliza RACF como gestor de seguridad externa, los nombres no deben contener el porcentaje. Si lo hacen, estos nombres no se incluyen en ninguna comprobación de seguridad cuando se utilizan perfiles genéricos de RACF .
- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ObjectName* el nombre de la cola creada. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si *ObjectName* es el nombre de una cola alias con TARGTYPE (TOPIC), primero se realiza una comprobación de seguridad en la cola alias con nombre; esto es normal cuando se utilizan colas alias. Cuando la comprobación de seguridad se completa correctamente, la llamada MQOPEN continuará y se comportará como una llamada MQOPEN en un MQOT_TOPIC; esto incluye realizar una comprobación de seguridad en el objeto de tema administrativo.
- Si *ObjectName* y *ObjectQMgrName* identifican una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, no debe haber también una definición de cola con el mismo nombre en el gestor de colas local. Si existe una definición de este tipo (una cola local, una cola alias, una cola remota o una cola modelo), la llamada falla con el código de razón MQRC_OBJECT_NOT_UNIQUE.
- Si el objeto que se está abriendo es una lista de distribución (es decir, *RecsPresent* está presente y es mayor que cero), *ObjectName* debe estar en blanco o la serie nula. Si no se cumple esta condición, la llamada falla con el código de razón MQRC_OBJECT_NAME_ERROR.
- Si *ObjectType* es MQOT_Q_MGR, se aplican reglas especiales; en este caso, el nombre debe estar completamente en blanco hasta el primer carácter nulo o el final del campo.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas en el que se define el objeto *ObjectName* . Los caracteres que son válidos en el nombre son los mismos que los de *ObjectName* (consulte “ObjectName (MQCHAR48)” en la página 489). Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ObjectType* es MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS o MQOT_Q_MGR, *ObjectQMgrName* debe estar en blanco o el nombre del gestor de colas local.
- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ObjectQMgrName* el nombre del gestor de colas en el que se crea la cola; este es el nombre del gestor de colas local. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si *ObjectName* es el nombre de una cola de clúster y *ObjectQMgrName* está en blanco, el destino de los mensajes enviados utilizando el descriptor de contexto de cola devuelto por la llamada MQOPEN

es elegido por el gestor de colas (o la salida de carga de trabajo de clúster, si hay uno instalado) de la forma siguiente:

  - Si se especifica MQOO_BIND_ON_OPEN, el gestor de colas selecciona una instancia determinada de la cola de clúster al procesar la llamada MQOPEN, y todos los mensajes colocados utilizando este descriptor de contexto de cola se envían a dicha instancia.
  - Si se especifica MQOO_BIND_NOT_FIXED, el gestor de colas puede elegir una instancia diferente de la cola de destino (que reside en un gestor de colas diferente del clúster) para cada llamada MQPUT sucesiva que utilice este descriptor de contexto de cola.

  Si la aplicación necesita enviar un mensaje a una instancia *específica* de una cola de clúster (es decir, una instancia de cola que reside en un gestor de colas determinado del clúster), la aplicación debe especificar el nombre de dicho gestor de colas en el campo *ObjectQMgrName* . Esto fuerza al gestor de colas local a enviar el mensaje al gestor de colas de destino especificado.

- Si *ObjectName* es el nombre de una cola compartida que es propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ObjectQMgrName* puede ser el nombre del grupo de compartición de colas, el nombre del gestor de colas local o en blanco; el mensaje se coloca en la misma cola, cualquiera que sea el valor especificado.

  Los grupos de compartición de colas sólo están soportados en z/OS.

- Si *ObjectName* es el nombre de una cola compartida que es propiedad de un grupo de compartición de colas remoto (es decir, un grupo de compartición de colas al que no pertenece el gestor de colas local), *ObjectQMgrName* debe ser el nombre del grupo de compartición de colas. Puede utilizar el nombre de un gestor de colas que pertenece a ese grupo, pero esto puede retrasar el mensaje si ese gestor de colas concreto no está disponible cuando el mensaje llega al grupo de compartición de colas.
- Si el objeto que se está abriendo es una lista de distribución (es decir, *RecsPresent* es mayor que cero), *ObjectQMgrName* debe estar en blanco o la serie nula. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_OBJECT_Q_MGR_NAME_ERROR.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

DynamicQName (MQCHAR48)

Es el nombre de una cola dinámica que va a crear la llamada MQOPEN. Esto sólo es relevante cuando *ObjectName* especifica el nombre de una cola modelo; en todos los demás casos, se ignora *DynamicQName* .

Los caracteres que son válidos en el nombre son los mismos que los de *ObjectName*, excepto que también es válido un asterisco. Un nombre que esté en blanco (o uno en el que sólo haya espacios en blanco antes del primer carácter nulo) no es válido si *ObjectName* es el nombre de una cola modelo.

Si el último carácter no en blanco del nombre es un asterisco (`*`), el gestor de colas sustituye el asterisco por una serie de caracteres que garantiza que el nombre generado para la cola es exclusivo en el gestor de colas local. Para permitir un número suficiente de caracteres para esto, el asterisco sólo es válido en las posiciones de la 1 a la 33. No debe haber caracteres que no sean espacios en blanco o un carácter nulo detrás del asterisco.

Es válido que el asterisco aparezca en la primera posición de carácter, en cuyo caso el nombre consta únicamente de los caracteres generados por el gestor de colas.

En z/OS, no utilice un nombre con el asterisco en la primera posición de carácter, ya que no puede haber comprobaciones de seguridad realizadas en una cola con un nombre completo que se genere automáticamente.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo lo determina el entorno:

- En z/OS, el valor es `'CSQ.*'`.

- En otras plataformas, el valor es `'AMQ.*'`.

El valor es una serie terminada en nulo en C y una serie rellenada en blanco en otros lenguajes de programación.

AlternateUserId (MQCHAR12)

Si especifica MQOO_ALTERNATE_USER_AUTHORITY para la llamada MQOPEN, o MQPMO_ALTERNATE_USER_AUTHORITY para la llamada MQPUT1 , este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura, en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación. Sin embargo, algunas comprobaciones se siguen realizando con el identificador de usuario actual (por ejemplo, comprobaciones de contexto).

Si se especifica MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY y este campo está totalmente en blanco hasta el primer carácter nulo o el final del campo, la apertura sólo puede realizarse correctamente si no se necesita autorización de usuario para abrir este objeto con las opciones especificadas.

Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY ni MQPMO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

Existen las siguientes diferencias en los entornos indicados:

- En z/OS, sólo se utilizan los primeros 8 caracteres de *`AlternateUserId`* para comprobar la autorización para la apertura. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario sólo debe contener caracteres permitidos por el gestor de seguridad externa.

  Si se especifica *`AlternateUserId`* para una cola, el gestor de colas puede utilizar posteriormente el valor cuando se colocan los mensajes. Si las opciones MQPMO_ * _CONTEXT especificadas en la llamada MQPUT o MQPUT1 hacen que el gestor de colas genere la información de contexto de identidad, el gestor de colas coloca *`AlternateUserId`* en el campo *`UserIdentifier`* en el MQMD del mensaje, en lugar del identificador de usuario actual.
- En otros entornos, *`AlternateUserId`* sólo se utiliza para comprobaciones de control de acceso en el objeto que se está abriendo. Si el objeto es una cola, *`AlternateUserId`* no afecta al contenido del campo *`UserIdentifier`* en el MQMD de los mensajes enviados utilizando ese manejador de cola.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

RecsPresent (MQLONG)

Es el número de registros de objeto MQOR que ha proporcionado la aplicación. Si este número es mayor que cero, indica que se está abriendo una lista de distribución, siendo *`RecsPresent`* el número de colas de destino de la lista. Una lista de distribución sólo puede contener un destino.

El valor de *`RecsPresent`* no debe ser menor que cero, y si es mayor que cero *`ObjectType`* debe ser MQOT_Q; la llamada falla con el código de razón MQRC_RECS_PRESENT_ERROR si no se cumplen estas condiciones.

En z/OS, este campo debe ser cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *`Version`* es menor que MQOD_VERSION_2.

Recuento de KnownDest(MQLONG)

Es el número de colas de la lista de distribución que se resuelven en colas locales y que se han abierto correctamente. El recuento no incluye las colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

Recuento de UnknownDest(MQLONG)

Es el número de colas de la lista de distribución que se resuelven en colas remotas y que se han abierto correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

Recuento de InvalidDest(MQLONG)

Es el número de colas de la lista de distribución que no se han podido abrir correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Nota: Si está presente, este campo sólo se establece si el parámetro **CompCode** de la llamada MQOPEN o MQPUT1 es MQCC_OK o MQCC_WARNING; no se establece si el parámetro **CompCode** es MQCC_FAILED.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

Desplazamiento de ObjectRec(MQLONG)

Es el desplazamiento en bytes del primer registro de objeto MQOR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ObjectRecOffset* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se está abriendo una lista de distribución, se debe proporcionar una matriz de uno o más registros de objeto MQOR para especificar los nombres de las colas de destino en la lista de distribución. Esto se puede hacer de una de dos maneras:

- Utilizando el campo de desplazamiento *ObjectRecOffset*.

  En este caso, la aplicación debe declarar su propia estructura que contenga un MQOD seguido de la matriz de registros MQOR (con tantos elementos de matriz como sean necesarios) y establecer *ObjectRecOffset* en el desplazamiento del primer elemento de la matriz desde el inicio del MQOD. Asegúrese de que este desplazamiento sea correcto y tenga un valor que se pueda acomodar dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Utilice *ObjectRecOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).
- Utilizando el campo de puntero *ObjectRecPtr*.

  En este caso, la aplicación puede declarar la matriz de estructuras MQOR por separado de la estructura MQOD y establecer *ObjectRecPtr* en la dirección de la matriz.

  Utilice *ObjectRecPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

Sea cual sea la técnica que elija, utilice uno de *ObjectRecOffset* y *ObjectRecPtr* ; la llamada falla con el código de razón MQRC_OBJECT_RECORDS_ERROR si ambos son cero, o ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Desplazamiento de ResponseRec(MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se está abriendo una lista de distribución, puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas que no se han podido abrir (campo*CompCode* en MQRR) y la razón de cada anomalía (campo*Reason* en MQRR). Los datos se devuelven en la matriz de registros de respuesta en el mismo orden en que aparecen los nombres de cola en la matriz de registros de objeto. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunas colas se han abierto correctamente mientras que otras han fallado, o todas han fallado pero por motivos diferentes); el código de razón MQRC_MULTIPLE_REASON de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro **Reason** de la llamada MQOPEN o MQPUT1 y los registros de respuesta no se establecen. Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Los registros de respuesta se pueden proporcionar de la misma forma que los registros de objeto, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en *ResponseRecPtr* ; para obtener detalles sobre cómo hacerlo, consulte "Desplazamiento de ObjectRec(MQLONG)" en la página 493. Sin embargo, no se puede utilizar más de uno de *ResponseRecOffset* y *ResponseRecPtr* ; la llamada falla con el código de razón MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

Para la llamada MQPUT1 , estos registros de respuesta se utilizan para devolver información sobre los errores que se producen cuando se envía el mensaje a las colas de la lista de distribución, así como los errores que se producen cuando se abren las colas. El código de terminación y el código de razón de la operación de transferencia para una cola sustituyen a los de la operación abierta para dicha cola sólo si el código de terminación de esta última era MQCC_OK o MQCC_WARNING.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

ObjectRecPtr (MQPTR)

Es la dirección del primer registro de objeto MQOR. *ObjectRecPtr* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Puede utilizar *ObjectRecPtr* o *ObjectRecOffset* para especificar los registros de objeto, pero no ambos; para la descripción del campo *ObjectRecOffset* , consulte "Desplazamiento de ObjectRec(MQLONG)" en la página 493. Si no utiliza *ObjectRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

ResponseRecPtr (MQPTR)

Es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice *ResponseRecPtr* o *ResponseRecOffset* para especificar los registros de respuesta, pero no ambos; para obtener detalles, consulte "Desplazamiento de ResponseRec(MQLONG)" en la página 493. Si no utiliza *ResponseRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

AlternateSecurityId (MQBYTE40)

Se trata de un identificador de seguridad que se pasa con *AlternateUserId* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas. *AlternateSecurityId* sólo se utiliza si:

- Se ha especificado MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN, o
- Se ha especificado MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 ,

y el campo *AlternateUserId* no está completamente en blanco hasta el primer carácter nulo o el final del campo.

En Windows, *AlternateSecurityId* se puede utilizar para proporcionar el identificador de seguridad (SID) de Windows que identifica de forma exclusiva el *AlternateUserId*. El SID de un usuario se puede obtener del sistema Windows utilizando la llamada de API de `LookupAccountName()` Windows .

En z/OS, este campo se ignora.

El campo *AlternateSecurityId* tiene la estructura siguiente:

- El primer byte es un entero binario que contiene la longitud de los datos significativos que siguen; el valor excluye el propio byte de longitud. Si no hay ningún identificador de seguridad, la longitud es cero.
- El segundo byte indica el tipo de identificador de seguridad que está presente; son posibles los valores siguientes:

  MQSIDT_NT_SECURITY_ID
  Identificador de seguridad de Windows .

  MQSIDT_NONE
  Sin identificador de seguridad.
- El tercer byte y los bytes subsiguientes hasta la longitud definida por el primer byte contienen el propio identificador de seguridad.
- Los bytes restantes en el campo se establecen en cero binario.

Puede utilizar el siguiente valor especial:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. El valor inicial de este campo es MQSID_NONE. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedQName (MQCHAR48)

Es el nombre de la cola de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQName* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una lista de distribución
- Una cola alias que hace referencia a un objeto de tema (consulte ResObjectString en su lugar).
- Una cola alias que se resuelve en un objeto de tema.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *ResolvedQName*. *ResolvedQMgrName* puede ser el nombre del gestor de colas local.

Si *ResolvedQName* es una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *ResolvedQName* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQMgrName* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el atributo de cola **DefBind** tiene el valor MQBND_BIND_NOT_FIXED)
- Una lista de distribución

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ObjectString (MQCHARV)

El campo ObjectString especifica el nombre de objeto largo.

Especifica el nombre de objeto largo que se va a utilizar. Sólo se hace referencia a este campo para determinados valores de *ObjectType*y se ignora para todos los demás valores. Consulte la descripción de *ObjectType* para obtener detalles de qué valores indican que se utiliza este campo.

Si *ObjectString* se especifica incorrectamente, de acuerdo con la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_OBJECT_STRING_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

SelectionString (MQCHARV)

Es la serie utilizada para proporcionar los criterios de selección utilizados al recuperar mensajes de una cola.

SelectionString no debe proporcionarse en los casos siguientes:

- Si *ObjectType* no es MQOT_Q
- Si la cola que se está abriendo no se está abriendo utilizando una de las opciones MQOO_BROWSE o MQOO_INPUT_ *

Si se proporciona *SelectionString* en estos casos, la llamada falla con el código de razón MQRC_SELECTOR_INVALID_FOR_TYPE.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura “MQCHARV-Serie de longitud variable” en la página 293 , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SELECTION_STRING_ERROR. La longitud máxima de *SelectionString* es MQ_SELECTOR_LENGTH.

El uso de *SelectionString* se describe en Selectores.

Serie ResObject(MQCHARV)

El campo ResObjectString es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en el campo *ObjectName* .

Este campo sólo se devuelve para temas y alias de cola que hacen referencia a un objeto de tema.

Si el nombre de objeto largo se proporciona en *ObjectString* y no se proporciona nada en *ObjectName*, el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si este campo se omite (es decir, ResObjectString.VSBufSize es cero), no se devolverá *ResObjectString* , pero la longitud se devolverá en ResObjectString.VSLength.

Si la longitud del almacenamiento intermedio (proporcionada en ResObjectStrng.VSBufSize) es más corta que la *ResObjectString*completa, la serie se truncará y devolverá tantos caracteres situados más a la derecha como quepa en el almacenamiento intermedio proporcionado.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_RES_OBJECT_STRING_ERROR.

ResolvedType (MQLONG)

El tipo del objeto (base) resuelto que se está abriendo.

Los valores posibles son:

MQOT_Q
: El objeto resuelto es una cola. Este valor se aplica cuando una cola se abre directamente o cuando se abre una cola alias que apunta a una cola.

MQOT_TOPIC
: El objeto resuelto es un tema. Este valor se aplica cuando se abre un tema directamente o cuando se abre una cola alias que apunta a un objeto de tema.

MQOT_NONE
: El tipo resuelto no es una cola ni un tema.

MQOR-Registro de objeto

Utilice la estructura MQOR para especificar el nombre de cola y el nombre de gestor de colas de una única cola de destino. MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1 .

Disponibilidad

La estructura MQOR está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Juego de caracteres y codificación

Los datos de MQOR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en la llamada MQOPEN, puede abrir una lista de colas; esta lista se denomina lista de distribución. Cada mensaje colocado utilizando el descriptor de contexto de cola devuelto por esa llamada MQOPEN se coloca en cada una de las colas de la lista, siempre que la cola se haya abierto correctamente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 505. Campos en MQOR para MQOR

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
ObjectName (nombre de objeto)	Ninguna	Serie nula o espacios en blanco
ObjectQMgrNombre (nombre de gestor de colas de objeto)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQOR_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQOR MyOR = {MQOR_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQOR

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
  MQCHAR48  ObjectName;      /* Object name */
  MQCHAR48  ObjectQMgrName;  /* Object queue manager name */
};
```

Declaración COBOL para MQOR

```
**   MQOR structure
  10 MQOR.
**    Object name
   15 MQOR-OBJECTNAME     PIC X(48).
**    Object queue manager name
   15 MQOR-OBJECTQMGRNAME PIC X(48).
```

Declaración PL/I para MQOR

```
dcl
 1 MQOR based,
```

```
  3 ObjectName     char(48), /* Object name */
  3 ObjectQMgrName char(48); /* Object queue manager name */
```

Declaración de Visual Basic para MQOR

```
Type MQOR
  ObjectName     As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

ObjectName (MQCHAR48)

Es el mismo que el campo *ObjectName* en la estructura MQOD (consulte MQOD para obtener detalles), excepto que:

- Debe ser el nombre de una cola.
- No debe ser el nombre de una cola modelo.

Siempre es un campo de entrada. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)

Es el mismo que el campo *ObjectQMgrName* en la estructura MQOD (consulte MQOD para obtener más detalles).

Siempre es un campo de entrada. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MQPD-Descriptor de propiedad

La estructura **MQPD** se utiliza para definir los atributos de una propiedad. La estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Disponibilidad

La estructura **MQPD** está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y para IBM MQ MQI clients.

Juego de caracteres y codificación

Los datos de **MQPD** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 506. Campos en MQPD		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
StrucId (identificador de estructura)	MQPD_STRUC_ID	'PD'
Versión (número de versión de estructura)	MQPD_VERSION_1	1
Options (opciones)	MQPD_NONE	0
Support (soporte necesario para la propiedad de mensaje)	MQPD_SUPPORT_OPTIONAL	0
Contexto (contexto de mensaje al que pertenece la propiedad)	MQPD_NO_CONTEXT	0
CopyOptions (opciones de copia a las que pertenece la propiedad)	MQCOPY_DEFAULT	0

Notas:

1. En el lenguaje de programación C, la variable de macro MQPD_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQPD MyPD = {MQPD_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este es el identificador de estructura; el valor debe ser:

MQPD_STRUC_ID
Identificador de la estructura del descriptor de propiedad.

Para el lenguaje de programación C, la constante **MQPD_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQPD_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQPD_STRUC_ID**.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQPD_VERSION_1
Estructura del descriptor de propiedad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQPD_CURRENT_VERSION
Versión actual de la estructura del descriptor de propiedad.

Siempre es un campo de entrada. El valor inicial de este campo es **MQPD_VERSION_1**.

Opciones (MQLONG)

El valor debe ser:

MQPD_NONE
No se ha especificado ninguna opción

Siempre es un campo de entrada. El valor inicial de este campo es MQPD_NONE.

Soporte (MQLONG)

Este campo describe qué nivel de soporte para la propiedad de mensaje es necesario para el gestor de colas, para que el mensaje que contiene esta propiedad se coloque en una cola. Esto sólo se aplica a las propiedades definidas por IBM MQ; el soporte para todas las demás propiedades es opcional.

El campo se establece automáticamente en el valor correcto cuando el gestor de colas conoce la propiedad definida por IBM MQ. Si no se reconoce la propiedad, se asigna MQPD_SUPPORT_OPTIONAL. Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por IBM MQque el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *Support* .

Cuando se establece una propiedad definida por IBM MQutilizando la llamada MQSETMP en un manejador de mensajes donde se ha establecido la opción MQCMHO_NO_VALIDATION, *Support* se convierte en un campo de entrada. Esto permite a una aplicación colocar una propiedad definida por IBM MQ, con el valor correcto, donde la propiedad no está soportada por el gestor de colas conectado, pero donde el mensaje está pensado para procesarse en otro gestor de colas.

El valor MQPD_SUPPORT_OPTIONAL siempre se asigna a propiedades que no son propiedades definidas por IBM MQ.

Si un gestor de colas IBM WebSphere MQ 7.0 , que da soporte a las propiedades de mensaje, recibe una propiedad que contiene un valor *Support* no reconocido, la propiedad se trata como si:

- Se ha especificado MQPD_SUPPORT_REQUIRED si alguno de los valores no reconocidos está contenido en MQPD_REJECT_UNSUP_MASK.
- Se ha especificado MQPD_SUPPORT_REQUIRED_IF_LOCAL si alguno de los valores no reconocidos están contenidos en MQPD_ACCEPT_UNSUP_IF_XMIT_MASK
- De lo contrario, se ha especificado MQPD_SUPPORT_OPTIONAL.

La llamada MQINQMP devuelve uno de los valores siguientes, o se puede especificar uno de los valores, cuando se utiliza la llamada MQSETMP en un manejador de mensajes donde se establece la opción MQCMHO_NO_VALIDATION:

MQPD_SUPPORT_OPTIONAL
Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas por IBM MQ.

MQPD_SUPPORT_REQUIRED
Se necesita soporte para la propiedad. El mensaje es rechazado por un gestor de colas que no da soporte a la propiedad definida por IBM MQ. La llamada MQPUT o MQPUT1 falla con el código de terminación MQCC_FAILED y el código de razón MQRC_UNSUPPORTED_PROPERTY.

MQPD_SUPPORT_REQUIRED_IF_LOCAL
El mensaje es rechazado por un gestor de colas que no soporta la propiedad definida por IBM MQsi el mensaje está destinado a una cola local. La llamada MQPUT o MQPUT1 falla con el código de terminación MQCC_FAILED y el código de razón MQRC_UNSUPPORTED_PROPERTY.

La llamada MQPUT o MQPUT1 se ejecuta correctamente si el mensaje está destinado a un gestor de colas remoto.

Es un campo de salida en la llamada MQINQMP y un campo de entrada en la llamada MQSETMP si el descriptor de mensaje se ha creado con la opción MQCMHO_NO_VALIDATION establecida. El valor inicial de este campo es MQPD_SUPPORT_OPTIONAL.

Contexto (MQLONG)

Describe a qué contexto de mensaje pertenece la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por IBM MQque el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *Context* .

Se puede especificar la opción siguiente:

CONTEXTO_USUARIO_MQPDD
La propiedad está asociada al contexto de usuario.

No se requiere ninguna autorización especial para poder establecer una propiedad asociada al contexto de usuario utilizando la llamada MQSETMP.

En un gestor de colas IBM WebSphere MQ 7.0 , una propiedad asociada con el contexto de usuario se guarda tal como se describe para MQOO_SAVE_ALL_CONTEXT. Una llamada MQPUT con MQPMO_PASS_ALL_CONTEXT especificado, hace que la propiedad se copie del contexto guardado en el nuevo mensaje.

Si la opción descrita anteriormente no es necesaria, se puede utilizar la opción siguiente:

MQPD_NO_CONTEXT
La propiedad no está asociada a un contexto de mensaje.

Un valor no reconocido se rechaza con un código *Reason* de MQRC_PD_ERROR

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQPD_NO_CONTEXT.

CopyOptions (MQLONG)

Describe en qué tipo de mensajes se debe copiar la propiedad. Es un campo sólo de salida para las propiedades IBM MQ definidas reconocidas; IBM MQ establece el valor adecuado.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida IBM MQ que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *CopyOptions* .

Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

REENVÍO de MQCOPY_FORWARD
Esta propiedad se copia en un mensaje que se está reenviando.

MQCOPY_PUBLISH
Esta propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQCOPY_REPLY
Esta propiedad se copia en un mensaje de respuesta.

INFORME de MQCOPY_REPORT
Esta propiedad se copia en un mensaje de informe.

MQCOPY_ALL
Esta propiedad se copia en todos los tipos de mensajes posteriores.

Opción predeterminada: Se puede especificar la siguiente opción para proporcionar el conjunto predeterminado de opciones de copia:

MQCOPY_DEFAULT
Esta propiedad se copia en un mensaje que se está reenviando, en un mensaje de informe o en un mensaje recibido por un suscriptor cuando se está publicando un mensaje.

Esto equivale a especificar la combinación de las opciones MQCOPY_FORWARD, más MQCOPY_REPORT, más MQCOPY_PUBLISH.

Si no se requiere ninguna de las opciones descritas anteriormente, utilice la opción siguiente:

MQCOPY_NONE
Utilice este valor para indicar que no se han especificado otras opciones de copia; mediante programación no existe ninguna relación entre esta propiedad y los mensajes posteriores. Esto siempre se devuelve para las propiedades del descriptor de mensaje.

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQCOPY_DEFAULT.

MQPMO-Opciones de transferencia de mensajes

La estructura MQPMO permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o se publican en los temas. La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1 .

Versión

La versión actual de MQPMO es MQPMO_VERSION_3. Determinados campos sólo están disponibles en determinadas versiones de MQPMO. Si necesita portar aplicaciones entre varios entornos, debe asegurarse de que la versión de MQPMO sea coherente en todos los entornos. Los campos que existen

sólo en determinadas versiones de la estructura se identifican como tales en este tema y en las descripciones de campo.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQPMO soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQPMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Juego de caracteres y codificación

Los datos de MQPMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 507. Campos en MQPMO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQPMO_STRUC_ID	'PMO¬'
Versión (número de versión de estructura)	MQPMO_VERSION_1	1
Opciones (opciones que controlan la acción de MQPUT y MQPUT1)	MQPMO_NONE	0
Tiempo de espera (reservado)	Ninguna	-1
Contexto (descriptor de contexto de objeto de la cola de entrada)	Ninguna	0
KnownDestCount (número de mensajes enviados correctamente a colas locales)	Ninguna	0
UnknownDestCount (número de mensajes enviados correctamente a colas remotas)	Ninguna	0
Recuento deInvalidDest (número de mensajes que no se han podido enviar)	Ninguna	0
ResolvedQName (nombre resuelto de la cola de destino)	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrNombre (nombre resuelto del gestor de colas de destino)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si *Version* es menor que MQPMO_VERSION_2.		
RecsPresent (número de registros de mensajes de colocación o registros de respuesta presentes)	Ninguna	0
PutMsgRecFields (distintivos que indican qué campos MQPMR están presentes)	MQPMRF_NONE	0
PutMsgRecOffset (desplazamiento del primer registro de mensaje de colocación desde el inicio de MQPMO)	Ninguna	0

Tabla 507. Campos en MQPMO (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
ResponseRecDesplazamiento (desplazamiento del primer registro de respuesta desde el inicio de MQPMO)	Ninguna	0
PutMsgRecPtr (dirección del primer registro de colocación de mensaje)	Ninguna	Puntero nulo o bytes nulos
ResponseRecPtr (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si *Version* es menor que MQPMO_VERSION_3.		
OriginalMsgHandle (descriptor de contexto de mensaje original)	MQHM_NONE	0
NewMsgHandle (nuevo manejador de mensajes)	MQHM_NONE	0
Acción (tipo de colocación que se realiza y relación entre el mensaje original especificado por el campo *OriginalMsgHandle* y el nuevo mensaje especificado por el campo *NewMsgHandle*)	MQACTP_NEW	0
PubLevel (nivel de suscripción de destino de la publicación)	Ninguna	9

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQPMO_DEFAULT contiene los valores listados en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura; el valor debe ser:

MQPMO_STRUC_ID
Identificador de la estructura de opciones de colocación de mensaje.

Para el lenguaje de programación C, también se define la constante MQPMO_STRUC_ID_ARRAY; tiene el mismo valor que MQPMO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQPMO_STRUC_ID.

Versión (MQLONG)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQPMO_VERSION_1
Estructura de opciones de colocación de Version-1 .

Esta versión está soportada en todos los entornos.

MQPMO_VERSION_2
Estructura de opciones de colocación de Version-2 .

Esta versión está soportada en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

MQPMO_VERSION_3
Estructura de opciones de colocación de mensaje Version-3 .

Esta versión está soportada en todos los entornos.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQPMO_CURRENT_VERSION
Versión actual de la estructura de opciones de colocación de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQPMO_VERSION_1.

Opciones de MQPMO (MQLONG)

El campo Opciones controla el funcionamiento de las llamadas **MQPUT** y **MQPUT1** .

Opción de ámbito. Puede especificar alguna o ninguna de las opciones MQPMO. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit). Se anotan las combinaciones que no son válidas; cualquier otra combinación es válida.

La siguiente opción controla el ámbito de las publicaciones enviadas:

MQPMO_SCOPE_QMGR

La publicación solo se envía a los suscriptores que se han suscrito a este gestor de colas. La publicación no se reenvía a ningún gestor de colas de publicación/suscripción remoto que haya realizado una suscripción a este gestor de colas, lo que altera temporalmente cualquier comportamiento que se haya establecido utilizando el atributo de tema PUBSCOPE.

Nota: Si no se establece, el ámbito de publicación viene determinado por el atributo de tema PUBSCOPE.

Opciones de publicación. Las opciones siguientes controlan la forma en que se publican los mensajes en un tema:

MQPMO_SUPPRESS_REPLYTO

Cualquier información especificada en los campos *ReplyToQ* y *ReplyToQMgr* del MQMD de esta publicación no se pasa a los suscriptores. Si esta opción se utiliza con una opción de informe que requiere un *ReplyToQ*, la llamada falla con MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETAIN

El gestor de colas debe retener la publicación que se está enviando. Esta retención permite a un suscriptor solicitar una copia de esta publicación después de la hora en que se publicó, utilizando la llamada MQSUBRQ. También permite que se envíe una publicación a las aplicaciones que realizan su suscripción después del momento en que se realizó esta publicación (a menos que elijan no enviarla utilizando la opción MQSO_NEW_PUBLICATIONS_ONLY). Si a una aplicación se le envía una publicación que se ha retenido, se indica mediante la propiedad de mensaje MQIsRetained de dicha publicación.

Sólo se puede retener una publicación en cada nodo del árbol de temas. Por lo tanto, si ya hay una publicación retenida para este tema, publicada por cualquier otra aplicación, se sustituye por esta publicación. Por lo tanto, es mejor evitar que más de un editor retenga mensajes sobre el mismo tema.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en el tema, en cuyo caso pueden coincidir varias publicaciones retenidas (en varios nodos del árbol de temas) y se pueden enviar varias publicaciones a la aplicación solicitante. Consulte la descripción de la llamada "MQSUBRQ-Solicitud de suscripción" en la página 807 para obtener más detalles.

Para obtener información sobre cómo interactúan las publicaciones retenidas con los niveles de suscripción, consulte Interceptación de publicaciones.

Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con MQRC_PUT_NOT_RETAINED.

MQPMO_NOT_OWN_SUBS

Indica al gestor de colas que la aplicación no desea enviar ninguna de sus publicaciones a las suscripciones que posee. Las suscripciones se consideran propiedad de la misma aplicación si los manejadores de conexión son los mismos.

MQPMO_WARN_IF_NO_SUBS_MATCHED

Si ninguna suscripción coincide con la publicación, devuelva un código de terminación (*CompCode*) de MQCC_WARNING y el código de razón MQRC_NO_SUBS_MATCHES.

Si la operación de colocación devuelve MQRC_NO_SUBS_MISMA, la publicación no se ha entregado a ninguna suscripción. Sin embargo, si se especifica la opción MQPMO_RETAIN en la operación de colocación, el mensaje se conserva y se entrega a cualquier suscripción coincidente definida posteriormente.

Una suscripción sobre el tema coincide con la publicación si se cumple alguna de las condiciones siguientes:

- El mensaje se entrega a la cola de suscripción
- El mensaje se habría entregado a la cola de suscripción, pero un problema con la cola significa que el mensaje no se puede transferir a la cola y, en consecuencia, se ha colocado en la cola de mensajes no entregados o se ha descartado.
- Se define una salida de direccionamiento que suprime la entrega del mensaje a la suscripción

Una suscripción sobre el tema no coincide con la publicación si se cumple alguna de las condiciones siguientes:

- La suscripción tiene una serie de selección que no coincide con la publicación
- La suscripción ha especificado la opción MQSO_PUBLICATION_ON_REQUEST
- La publicación no se entrega porque se ha especificado la opción MQPMO_NOT_OWN_SUBS en la operación de colocación y la suscripción coincide con la identidad del publicador

Opciones de punto de sincronismo. Las opciones siguientes están relacionadas con la participación de la llamada MQPUT o MQPUT1 dentro de una unidad de trabajo:

MQPMO_SYNCPOINT
La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

Si no se especifican MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT, la inclusión de la solicitud de colocación en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas y no por el entorno que ejecuta la aplicación. En z/OS, la solicitud de colocación está dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de colocación no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT explícitamente.

No especifique MQPMO_SYNCPOINT con MQPMO_NO_SYNCPOINT.

MQPMO_NO_SYNCPOINT
La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje está disponible inmediatamente y no se puede suprimir restituir una unidad de trabajo.

Si no se especifican MQPMO_NO_SYNCPOINT y MQPMO_SYNCPOINT, la inclusión de la solicitud de colocación en protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas y no por el entorno que ejecuta la aplicación. En z/OS, la solicitud de colocación está dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de colocación no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT explícitamente.

No especifique MQPMO_NO_SYNCPOINT con MQPMO_SYNCPOINT.

Opciones de identificador de mensaje e identificador de correlación. Las opciones siguientes solicitan al gestor de colas que genere un nuevo identificador de mensaje o identificador de correlación:

MQPMO_NEW_MSG_ID
El gestor de colas sustituye el contenido del campo *MsgId* en MQMD por un nuevo identificador de mensaje. Este identificador de mensaje se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

La opción MQPMO_NEW_MSG_ID también se puede especificar cuando el mensaje se transfiere a una lista de distribución; consulte la descripción del campo *MsgId* en la estructura MQPMR para obtener detalles.

El uso de esta opción libera a la aplicación de la necesidad de restablecer el campo *MsgId* a MQMI_NONE antes de cada llamada MQPUT o MQPUT1 .

MQPMO_NEW_CORREL_ID
El gestor de colas sustituye el contenido del campo *CorrelId* en MQMD por un nuevo identificador de correlación. Este identificador de correlación se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

La opción MQPMO_NEW_CORREL_ID también se puede especificar cuando el mensaje se coloca en una lista de distribución; consulte la descripción del campo *CorrelId* en la estructura MQPMR para obtener detalles.

MQPMO_NEW_CORREL_ID es útil en situaciones en las que la aplicación requiere un identificador de correlación exclusivo.

Opciones de grupo y segmento. Las opciones siguientes están relacionadas con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Lea las definiciones siguientes para ayudarle a comprender la opción.

Atención: No puede utilizar los mensajes segmentados ni agrupados con la publicación/suscripción.

Mensaje físico
Es la unidad de información más pequeña que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Generalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (campo*MsgId* en MQMD), aunque el gestor de colas no los impone.

Mensaje lógico
Un mensaje lógico es una única unidad de información de aplicación sólo para plataformas que no son de z/OS . En ausencia de restricciones del sistema, un mensaje lógico es el mismo que un mensaje físico. Pero cuando los mensajes lógicos son extremadamente grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados *segmentos*.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (campo*GroupId* en MQMD) y el mismo número de secuencia de mensaje (campo*MsgSeqNumber* en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento (campo*Offset* en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Debido a que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo, aunque en este caso sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos para los que la aplicación emisora ha inhibido la segmentación tienen un identificador de grupo nulo (MQGI_NONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes
Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes para el número de secuencia de mensaje, que es un entero en el rango de 1 a *n*, donde *n* es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de *n* mensajes físicos en el grupo.

MQPMO_LOGICAL_ORDER
Esta opción indica al gestor de colas cómo la aplicación coloca los mensajes en grupos y segmentos de mensajes lógicos. Sólo se puede especificar en la llamada MQPUT; no es válida en la llamada MQPUT1.

Si se especifica MQPMO_LOGICAL_ORDER, indica que la aplicación utiliza llamadas MQPUT sucesivas para:

1. Poner los segmentos de cada mensaje lógico en el orden de desplazamiento de segmento creciente, empezando desde 0, sin huecos.
2. Poner todos los segmentos en un mensaje lógico antes de poner los segmentos en el siguiente mensaje lógico.
3. Poner los mensajes lógicos de cada grupo de mensajes en el orden de número de secuencia de mensaje creciente, empezando desde 1, sin huecos. IBM MQ incrementa el número de secuencia de mensaje automáticamente.
4. Poner todos los mensajes lógicos en un grupo de mensajes antes de poner los mensajes lógicos en el siguiente grupo de mensajes.

Para obtener información detallada sobre MQPMO_LOGICAL_ORDER, consulte Ordenación lógica y física

Opciones de contexto. Las opciones siguientes controlan el proceso del contexto de mensaje:

MQPMO_NO_CONTEXT
Tanto el contexto de identidad como el contexto de origen se establecen para indicar que no hay contexto. Esto significa que los campos de contexto en MQMD se establecen en:

- Espacios en blanco para campos de caracteres
- Nulos para campos de bytes
- Ceros para campos numéricos

MQPMO_DEFAULT_CONTEXT
El mensaje debe tener asociada la información de contexto predeterminada, tanto para la identidad como para el origen. El gestor de colas establece los campos de contexto en el descriptor de mensaje como se indica a continuación:

Tabla 508. Valores de información de contexto predeterminados para campos MQMD

Campo de MQMD	Valor utilizado
UserIdentifier	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
AccountingToken	Determinado por el entorno si es posible; en caso contrario, establecido en MQACT.
ApplIdentityData	Establecido en blancos.
PutApplType	Determinado a partir del entorno.
PutApplName	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
PutDate	Establézcalo en la fecha en la que se coloca el mensaje.
PutTime	Establézcalo en la hora en que se coloca el mensaje.
ApplOriginData	Establecido en blancos.

Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Estos son los valores y acciones predeterminados si no se especifica ninguna opción de contexto.

MQPMO_PASS_IDENTITY_CONTEXT
El mensaje tiene que tener información de contexto asociada. El contexto de identidad se toma del descriptor de contexto de cola especificado en el campo *Context* . El gestor de colas genera la información de contexto de origen de la misma forma que para MQPMO_DEFAULT_CONTEXT (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_IDENTITY_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_PASS_IDENTITY_CONTEXT.

MQPMO_PASS_ALL_CONTEXT
El mensaje tiene que tener información de contexto asociada. El contexto se toma del descriptor de contexto de cola especificado en el campo *Context* . Para obtener más información sobre el contexto de mensaje, consulte Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_ALL_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_PASS_ALL_CONTEXT.

MQPMO_SET_IDENTITY_CONTEXT
El mensaje tiene que tener información de contexto asociada. La aplicación especifica el contexto de identidad en la estructura MQMD. El gestor de colas genera la información de contexto de origen de la misma forma que para MQPMO_DEFAULT_CONTEXT (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_IDENTITY_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_SET_IDENTITY_CONTEXT.

MQPMO_SET_ALL_CONTEXT
El mensaje tiene que tener información de contexto asociada. La aplicación especifica la identidad, el origen y el contexto de usuario en la estructura MQMD. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_ALL_CONTEXT. Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_SET_ALL_CONTEXT.

Sólo puede especificar una de las opciones de contexto MQPMO_ * _CONTEXT. Si no especifica ninguno, se presupone MQPMO_DEFAULT_CONTEXT.

Opciones de propiedad. La siguiente opción está relacionada con las propiedades del mensaje:

MQPMO_MD_FOR_OUTPUT_ONLY
El parámetro de descriptor de mensaje sólo debe utilizarse para que la salida devuelva el descriptor de mensaje del mensaje que se ha colocado. Los campos de descriptor de mensaje asociados con los campos *NewMsgHandle*, *OriginalMsgHandle*, o ambos, de la estructura **MQPMO** se deben utilizar para la entrada.

Si no se proporciona un manejador de mensajes válido, la llamada falla con el código de razón **MQRC_MD_ERROR**.

Opciones de respuesta de colocación. Las opciones siguientes controlan la respuesta devuelta a una llamada MQPUT o MQPUT1 . Sólo puede especificar una de estas opciones. Si no se especifican MQPMO_ASYNC_RESPONSE y MQPMO_SYNC_RESPONSE, se presupone MQPMO_RESPONSE_AS_Q_DEF o MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_ASYNC_RESPONSE
La opción MQPMO_ASYNC_RESPONSE solicita que se complete una operación MQPUT o MQPUT1 sin que la aplicación espere a que el gestor de colas complete la llamada. La utilización de esta opción puede mejorar el rendimiento de la mensajería, especialmente para las aplicaciones que utilizan enlaces de cliente. Una aplicación puede comprobar periódicamente, utilizando el verbo MQSTAT, si se ha producido un error durante cualquier llamada asíncrona anterior.

Con esta opción, solo se garantiza que se completen los campos siguientes en el MQMD;

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Además, si se especifica MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID como opciones, o ambos, el MsgId y el CorrelId devueltos también se completan. (MQPMO_NEW_MSG_ID puede especificarse implícitamente especificando un campo MsgId en blanco).

Sólo se completan los campos especificados anteriores. Otra información que normalmente se devolvería en la estructura MQMD o MQPMO no está definida.

Al solicitar una respuesta de transferencia asíncrona para MQPUT1, el nombre ResolvedQName y ResolvedQMgrdevuelto en la estructura MQOD no están definidos.

Cuando se solicita una respuesta de transferencia asíncrona para MQPUT o MQPUT1, un código CompCode y una razón MQCC_OK y MQRC_NONE no significan necesariamente que el mensaje se haya transferido correctamente a una cola. Al desarrollar una aplicación MQI que utiliza la respuesta de colocación asíncrona y requiere confirmación de que los mensajes se han colocado en una cola, debe comprobar los códigos CompCode y Reason de las operaciones de colocación y también utilizar MQSTAT para consultar información de errores asíncronos.

Aunque el éxito o el error de cada llamada MQPUT o MQPUT1 individual no se devuelven inmediatamente, el primer error que se ha producido bajo una llamada asíncrona se puede determinar más adelante mediante una llamada a MQSTAT.

Si un mensaje persistente bajo punto de sincronismo no se puede entregar utilizando la respuesta de transferencia asíncrona e intenta confirmar la transacción, la confirmación falla y la transacción se restituye con un código de terminación de MQCC_FAILED y una razón de MQRC_BACKED_OUT. La aplicación puede realizar una llamada a MQSTAT para determinar la causa de una anomalía anterior de MQPUT o MQPUT1 .

MQPMO_SYNC_RESPONSE

La especificación de este tipo de respuesta put garantiza que la operación MQPUT o MQPUT1 se emita siempre de forma síncrona. Si la operación de colocación es satisfactoria, se completan todos los campos de MQMD y MQPMO.

Esta opción garantiza una respuesta síncrona independientemente del valor de respuesta de colocación predeterminado definido en la cola o el objeto de tema.

MQPMO_RESPONSE_AS_Q_DEF

Si se especifica este valor para una llamada MQPUT, el tipo de respuesta de colocación utilizado se toma del valor DEFPRESP especificado en la cola cuando la aplicación la abrió por primera vez.

- Si la cola es una cola de clúster, y este valor se especifica para una llamada MQPUT, el tipo de respuesta de colocación utilizado se toma del atributo **DEFPRESP** definido en el gestor de colas de *destino* que es propietario de la instancia concreta de la cola en la que se coloca el mensaje.

  Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también ¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?
- Si la cola no es una cola de clúster, y este valor se especifica para una llamada MQPUT, el tipo de respuesta de colocación utilizado se obtiene del atributo **DEFPRESP** definido en el gestor de colas *local* , incluso si el gestor de colas de destino es remoto.

Si una aplicación cliente está conectada a un gestor de colas en un nivel anterior a IBM WebSphere MQ 7.0, se comporta como si se hubiera especificado MQPMO_SYNC_RESPONSE.

Si se especifica esta opción para una llamada MQPUT1 , el valor del atributo DEFPRESP no se conoce antes de que se envíe la solicitud al servidor. De forma predeterminada, si la llamada MQPUT1 utiliza MQPMO_SYNCPOINT, se comporta como para MQPMO_ASYNC_RESPONSE, y si utiliza MQPMO_NO_SYNCPOINT, se comporta como para MQPMO_SYNC_RESPONSE. Sin embargo, puede alterar temporalmente este comportamiento predeterminado estableciendo la propiedad Put1DefaultAlwaysSync en el archivo de configuración del cliente, consulte Stanza CHANNELS del archivo de configuración del cliente.

MQPMO_RESPONSE_AS_TOPIC_DEF
MQPMO_RESPONSE_AS_TOPIC_DEF es un sinónimo de MQPMO_RESPONSE_AS_Q_DEF para su uso con objetos de tema.

Otras opciones. Las opciones siguientes controlan la comprobación de autorización, lo que sucede cuando el gestor de colas se está desactivando temporalmente y la resolución de nombres de colas y gestores de colas:

MQPMO_ALTERNATE_USER_AUTHORITY
MQPMO_ALTERNATE_USER_AUTHORITY indica que el campo *AlternateUserId* del parámetro **ObjDesc** de la llamada MQPUT1 contiene un identificador de usuario que se debe utilizar para validar la autorización para colocar mensajes en la cola. La llamada sólo puede realizarse correctamente si *AlternateUserId* tiene autorización para abrir la cola con las opciones especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación tiene autorización para hacerlo. (Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.)

Esta opción sólo es válida con la llamada MQPUT1 .

MQPMO_FAIL_IF_QUIESCING
Esta opción fuerza que la llamada MQPUT o MQPUT1 falle si el gestor de colas está en estado de desactivación temporal.

En z/OS, esta opción también fuerza que la llamada MQPUT o MQPUT1 falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

La llamada devuelve el código de terminación MQCC_FAILED con el código de razón MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.

MQPMO_RESOLVE_LOCAL_Q
Utilice esta opción para rellenar *ResolvedQName* en la estructura MQPMO con el nombre de la cola local a la que se coloca el mensaje y *ResolvedQMgrName* con el nombre del gestor de colas local que aloja la cola local. Para obtener más información sobre MQPMO_RESOLVE_LOCAL_Q, consulte el tema MQOO_RESOLVE_LOCAL_Q.

Si tiene autorización para transferir a una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQPUT; no es necesaria ninguna autorización especial.

Opción predeterminada. Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQPMO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQPMO_NONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

MQPMO_NONE es un campo de entrada. El valor inicial del campo *Options* es MQPMO_NONE.

Tiempo de espera (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es `-1`.

Contexto (MQHOBJ)

Si se especifica MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT, este campo debe contener el descriptor de contexto de la cola de entrada del que se toma la información de contexto que se va a asociar con el mensaje que se va a colocar.

Si no se especifica MQPMO_PASS_IDENTITY_CONTEXT ni MQPMO_PASS_ALL_CONTEXT, este campo se ignora.

Este es un campo de entrada. El valor inicial de este campo es `0`.

Recuento de KnownDest(MQLONG)

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que son colas locales. El recuento no incluye los mensajes enviados a colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

Recuento de UnknownDest(MQLONG)

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que se resuelven en colas remotas. Los mensajes que el gestor de colas retiene temporalmente en formato de lista de distribución cuentan como el número de destinos individuales que contienen esas listas de distribución. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

Recuento de InvalidDest(MQLONG)

Es el número de mensajes que no se han podido enviar a las colas de la lista de distribución. El recuento incluye las colas que no se han podido abrir, así como las colas que se han abierto correctamente pero para las que la operación de colocación ha fallado. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Nota: Este campo se establece si el parámetro **CompCode** de la llamada MQPUT o MQPUT1 es MQCC_OK o MQCC_WARNING; se puede establecer si el parámetro **CompCode** es MQCC_FAILED, pero no se basa en esto en el código de aplicación.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

ResolvedQName (MQCHAR48)

Es el nombre de la cola de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *ResolvedQName*, y puede ser el nombre del gestor de colas local.

Si *ResolvedQName* es una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *ResolvedQName* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

RecsPresent (MQLONG)

Es el número de registros de mensajes de colocación MQPMR o registros de respuesta MQRR que ha proporcionado la aplicación. Este número puede ser mayor que cero sólo si el mensaje se está colocando en una lista de distribución. Los registros de mensajes de colocación y los registros de respuesta son opcionales; la aplicación no necesita proporcionar ningún registro, o puede optar por proporcionar registros de un solo tipo. Sin embargo, si la aplicación proporciona registros de ambos tipos, debe proporcionar registros *RecsPresent* de cada tipo.

No es necesario que el valor de *RecsPresent* sea el mismo que el número de destinos de la lista de distribución. Si se proporcionan demasiados registros, el exceso no se utiliza; si se proporcionan demasiados pocos registros, se utilizan los valores predeterminados para las propiedades de mensaje para los destinos que no tienen registros de mensajes de colocación (consulte *PutMsgRecOffset*).

Si *RecsPresent* es menor que cero, o es mayor que cero pero el mensaje no se está colocando en una lista de distribución, la llamada falla con el código de razón MQRC_RECS_PRESENT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecFields (MQLONG)

Este campo contiene distintivos que indican qué campos MQPMR están presentes en los registros de mensajes de colocación proporcionados por la aplicación. Utilice *PutMsgRecFields* sólo cuando el mensaje se transfiera a una lista de distribución. El campo se ignora si *RecsPresent* es cero, o si *PutMsgRecOffset* y *PutMsgRecPtr* son cero.

Para los campos que están presentes, el gestor de colas utiliza para cada destino los valores de los campos del registro de mensajes de colocación correspondiente. Para los campos que están ausentes, el gestor de colas utiliza los valores de la estructura MQMD.

Utilice uno o varios de los distintivos siguientes para indicar qué campos están presentes en los registros de mensajes de colocación:

MQPMRF_MSG_ID
El campo de identificador de mensaje está presente.

ID MQPMRF_CORREL_ID
El campo de identificador de correlación está presente.

MQPMRF_ID_grupo
El campo identificador de grupo está presente.

MQPMRF_FEEDBACK
El campo de comentarios está presente.

MQPMRF_ACCOUNTING_TOKEN
El campo de señal de contabilidad está presente.

Si especifica este distintivo, especifique MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el campo *Options* ; si no se cumple esta condición, la llamada falla con el código de razón MQRC_PMO_RECORD_FLAGS_ERROR.

Si no hay campos MQPMR presentes, se puede especificar lo siguiente:

MQPMRF_NONE
No hay campos de registro de colocación de mensaje.

Si se especifica este valor, *RecsPresent* debe ser cero, o bien *PutMsgRecOffset* y *PutMsgRecPtr* deben ser cero.

MQPMRF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

Si *PutMsgRecFields* contiene distintivos que no son válidos, o se proporcionan registros de colocación de mensajes pero *PutMsgRecFields* tiene el valor MQPMRF_NONE, la llamada falla con el código de razón MQRC_PMO_RECORD_FLAGS_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQPMRF_NONE. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de mensajes de colocación MQPMR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PutMsgRecOffset* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando el mensaje se transfiere a una lista de distribución, se puede proporcionar una matriz de uno o más registros de mensajes de colocación MQPMR para especificar determinadas propiedades del mensaje para cada destino individualmente; estas propiedades son:

- Identificador del mensaje
- Identificador de correlación
- Identificador de grupo
- Valor de comentarios
- Señal de contabilidad

No es necesario especificar todas estas propiedades, pero independientemente del subconjunto que elija, especifique los campos en el orden correcto. Consulte la descripción de la estructura MQPMR para obtener más detalles.

Normalmente, debe haber tantos registros de mensajes de colocación como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cada registro de mensajes de colocación proporciona las propiedades de mensaje para la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben tener asignados registros de mensajes en las posiciones adecuadas de la matriz, aunque las propiedades del mensaje se ignoran en este caso.

El número de registros de mensajes de colocación puede diferir del número de registros de objeto. Si hay menos registros de mensaje de colocación que registros de objeto, las propiedades de mensaje para los destinos que no tienen registros de mensaje de colocación se toman de los campos correspondientes en el MQMD del descriptor de mensaje. Si hay más registros de mensajes de colocación que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible acceder a ellos). Los registros de mensajes de colocación son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Proporcione los registros de mensajes de colocación de forma similar a los registros de objeto en MQOD, ya sea especificando un desplazamiento en *PutMsgRecOffset*o especificando una dirección en *PutMsgRecPtr* ; para obtener detalles sobre cómo hacerlo, consulte el campo *ObjectRecOffset* descrito en“MQOD-Descriptor de objeto” en la página 483.

No se puede utilizar más de uno de *PutMsgRecOffset* y *PutMsgRecPtr* ; la llamada falla con el código de razón MQRC_PUT_MSG_RECORDS_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Desplazamiento de ResponseRec(MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Al colocar el mensaje en una lista de distribución, puede proporcionar una matriz de uno o varios registros de respuesta MQRR para identificar las colas a las que no se ha enviado correctamente el mensaje

(campo*CompCode* en MQRR) y la razón de cada anomalía (campo*Reason* en MQRR). Es posible que el mensaje no se haya enviado porque la cola no se ha podido abrir o porque la operación de colocación ha fallado. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunos mensajes se han enviado correctamente mientras que otros han fallado, o todos han fallado pero por razones diferentes); el código de razón MQRC_MULTIPLE_REASON de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro **Reason** de la llamada MQPUT o MQPUT1 y los registros de respuesta no se establecen.

Normalmente, hay tantos registros de respuesta como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cuando es necesario, cada registro de respuesta se establece en el código de terminación y el código de razón para la colocación en la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de respuesta asignados en las posiciones adecuadas de la matriz, aunque se establecen en el código de terminación y el código de razón resultantes de la operación de apertura, en lugar de en la operación de colocación.

El número de registros de respuesta puede diferir del número de registros de objeto. Si hay menos registros de respuesta que registros de objeto, es posible que la aplicación no pueda identificar todos los destinos para los que ha fallado la operación de colocación, o las razones de las anomalías. Si hay más registros de respuesta que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible acceder a ellos). Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Proporcione los registros de respuesta de forma similar a los registros de objeto en MQOD, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en *ResponseRecPtr* ; para obtener detalles sobre cómo hacerlo, consulte el campo *ObjectRecOffset* descrito en “MQOD-Descriptor de objeto” en la página 483. Sin embargo, no utilice más de uno de *ResponseRecOffset* y *ResponseRecPtr* ; la llamada falla con el código de razón MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

Para la llamada MQPUT1 , este campo debe ser cero. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR)

Es la dirección del primer registro de mensaje de colocación MQPMR. Utilice *PutMsgRecPtr* sólo cuando el mensaje se transfiera a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Puede utilizar *PutMsgRecPtr* o *PutMsgRecOffset* para especificar los registros de mensajes de colocación, pero no ambos; para obtener detalles, consulte “PutMsgRecOffset (MQLONG)” en la página 518. Si no utiliza *PutMsgRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

ResponseRecPtr (MQPTR)

Es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice *ResponseRecPtr* o *ResponseRecOffset* para especificar los registros de respuesta, pero no ambos; para obtener detalles, consulte “Desplazamiento de ResponseRec(MQLONG)” en la página 518. Si no utiliza *ResponseRecPtr* , establézcalo en el puntero nulo o en bytes nulos.

Para la llamada MQPUT1 , este campo debe ser el puntero nulo o bytes nulos. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

Manejador OriginalMsg(MQHMSG)

Se trata de un descriptor de contexto opcional para un mensaje. Es posible que se haya recuperado anteriormente de una cola. El uso de este descriptor de contexto está sujeto al valor del campo *Action* ; consulte también NewMsgHandle.

La llamada **MQPUT** o **MQPUT1** no alterará el contenido del manejador de mensajes original.

Este es un campo de entrada. El valor inicial de este campo es **MQHM_NONE**. Este campo se ignora si la versión es menor que **MQPMO_VERSION_3**.

Manejador NewMsg(MQHMSG)

Se trata de un descriptor de contexto opcional para el mensaje que se coloca sujeto al valor del campo Acción. Define las propiedades del mensaje y altera temporalmente los valores de *OriginalMsgHandle*, si se especifica.

Al volver de la llamada **MQPUT** o **MQPUT1** , el contenido del descriptor de contexto refleja el mensaje que se ha colocado realmente.

Este es un campo de entrada. El valor inicial de este campo es **MQHM_NONE**. Este campo se ignora si la versión es menor que **MQPMO_VERSION_3**.

Acción (MQLONG)

Especifica el tipo de colocación que se está realizando y la relación entre el mensaje original especificado por el campo de manejador OriginalMsgy el nuevo mensaje especificado por el campo de manejador NewMsg. El gestor de colas elige las propiedades del mensaje de acuerdo con el valor de la acción especificada.

Puede optar por proporcionar el contenido del descriptor de mensaje utilizando el parámetro MsgDesc en las llamadas MQPUT o MQPUT1 . De forma alternativa, es posible no proporcionar el parámetro MsgDesc o especificar que es sólo de salida incluyendo MQPMO_MD_FOR_OUTPUT_ONLY en el campo Opciones de la estructura MQPMO.

Si no se proporciona el parámetro MsgDesc , o si se especifica que sea sólo de salida, el descriptor de mensaje para el nuevo mensaje se llena a partir de los campos de manejador de mensajes de MQPMO, de acuerdo con las reglas descritas en este tema.

El valor de contexto y las actividades de paso descritas en Control de la información de contexto entran en vigor después de que se haya compuesto el descriptor de mensaje.

Si se especifica un valor de acción incorrecto, la llamada falla con el código de razón MQRC_ACTION_ERROR.

Se puede especificar cualquiera de las acciones siguientes:

MQACTP_NEW
: Se está colocando un nuevo mensaje y el programa no está especificando ninguna relación con un mensaje anterior. El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Los campos de descriptor de mensaje establecidos explícitamente en el nuevo descriptor de mensaje tienen prioridad sobre los del descriptor de mensaje original.

Los datos de mensaje se toman del parámetro de almacenamiento intermedio MQPUT o MQPUT1 .

MQACTP_FORWARD

Se está reenviando un mensaje recuperado anteriormente. El descriptor de mensaje original especifica el mensaje que se ha recuperado anteriormente.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Los campos de descriptor de mensaje establecidos explícitamente en el nuevo descriptor de mensaje tienen prioridad sobre los del descriptor de mensaje original.
- Si se especifica MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID en el MQPMO de MQPMO.Options, a continuación, se respetan.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_FORWARD en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos de mensaje que se van a reenviar se toman del parámetro de almacenamiento intermedio MQPUT o MQPUT1 .

MQACTP_REPLY

Se está realizando una respuesta a un mensaje recuperado anteriormente. El descriptor de mensaje original especifica el mensaje que se ha recuperado anteriormente.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Optionsy, a continuación, los campos de descriptor de mensaje inicial se seleccionan de la forma siguiente:

Tabla 509. Transformación de manejador de mensajes de respuesta	
Campo de MQMD	**Valor utilizado**
Informe	Si MQRO_PASS_DESCARD_AND_EXPIRE y MQRO_DISCARD_MSG están establecidos: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPLY
Caducidad	Si MQRO_PASS_DESCARD_AND_EXPIRE está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
Comentarios	MQFB_NONE
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje de lo contrario, si se establece MQRO_PASS_MSG_ID: Copiado del mensaje de entrada de lo contrario MQMI_NONE
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación else si se establece MQRO_COPY_MSG_ID_TO_CORREL_ID: Copiado del campo MsgId del mensaje de entrada else si se establece MQRO_PASS_CORREL_ID: Copiado del campo CorrelId del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Espacios en blanco
GroupId	MQGI_NONE
MsgSeqNumber	1
Desplazamiento	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- A continuación, el descriptor de mensaje lo modifica el nuevo descriptor de mensaje: los campos de descriptor de mensaje establecidos explícitamente como propiedades en el nuevo descriptor de mensaje tienen prioridad sobre los campos de descriptor de mensaje tal como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_REPLY en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos de mensaje que se van a reenviar se toman del parámetro de almacenamiento intermedio MQPUT/MQPUT1 .

MQACTP_REPORT

Se está generando un informe como resultado de un mensaje recuperado anteriormente. El descriptor de contexto de mensaje original especifica el mensaje que hace que se genere el informe.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options y, a continuación, los campos del descriptor de mensaje inicial se seleccionan de la forma siguiente:

Tabla 510. Transformación de manejador de mensajes de informe

Campo de MQMD	**Valor utilizado**
Informe	Si MQRO_PASS_DESCARD_AND_EXPIRE y MQRO_DISCARD_MSG están establecidos: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPORT
Caducidad	Si MQRO_PASS_DESCARD_AND_EXPIRE está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje de lo contrario, si se establece MQRO_PASS_MSG_ID: Copiado del mensaje de entrada de lo contrario MQMI_NONE

Tabla 510. Transformación de manejador de mensajes de informe (continuación)	
Campo de MQMD	**Valor utilizado**
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación else si se establece MQRO_COPY_MSG_ID_TO_CORREL_ID: Copiado del campo MsgId del mensaje de entrada else si se establece MQRO_PASS_CORREL_ID: Copiado del campo CorrelId del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Espacios en blanco
OriginalLength	Establézcalo en *BufferLength*

- A continuación, el descriptor de mensaje lo modifica el nuevo descriptor de mensaje: los campos de descriptor de mensaje establecidos explícitamente como propiedades en el nuevo descriptor de mensaje tienen prioridad sobre los campos de descriptor de mensaje tal como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_REPORT en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

El campo Comentarios en el MQMD resultante representa el informe que se va a generar. Un valor de retorno de MQFB_NONE hace que la llamada MQPUT o MQPUT1 falle con el código de razón MQRC_FEEDBACK_ERROR.

Para elegir los datos de usuario del mensaje de informe, IBM MQ consulta los campos Informe y Comentarios en el MQMD resultante, y los parámetros Buffer y BufferLength de la llamada MQPUT o MQPUT1 .

- Si los comentarios son MQFB_COA, MQFB_COD o MQFB_EXPIRATION, se inspecciona el valor del informe.
- Si se cumple alguno de los casos siguientes, se utilizan los datos de mensaje completos del almacenamiento intermedio para una longitud de BufferLength .
  - Los comentarios son MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_FULL_DATA
  - Los comentarios son MQFB_COD y el informe contiene MQRO_COD_WITH_FULL_DATA
  - Los comentarios son MQFB_COA y el informe contiene MQRO_COA_WITH_FULL_DATA
- Si se cumple alguno de los casos siguientes, se utilizan los primeros 100 bytes del mensaje (o BufferLength si es inferior a 100) del almacenamiento intermedio
  - Los comentarios son MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_DATA

  - Los comentarios son MQFB_COD y el informe contiene MQRO_COD_WITH_DATA
  - Los comentarios son MQFB_COA y el informe contiene MQRO_COA_WITH_DATA
- Si los comentarios son MQFB_EXPIRATION, MQFB_COD o MQFB_COA, y el informe no contiene las opciones * _WITH_FULL_DATA o * _WITH_DATA relevantes para ese valor de Feeback, no se incluirán datos de usuario con el mensaje.
- Si Feedback toma un valor diferente de los listados anteriormente, Buffer y BufferLength se utilizan de forma normal.

La derivación de los datos de usuario descritos en la lista anterior también se muestra en la tabla siguiente:

Tabla 511. Origen de datos de usuario

	MQFB_COA	**MQFB_COD**	**MQFB_EXPIRATION**
MQRO_EXPIRATION _WITH_FULL_DATA	Ninguna	Ninguna	Almacenamiento intermedio (Bufferlength)
MQRO_COD_WITH _FULL_DATA	Ninguna	Almacenamiento intermedio (Bufferlength)	Ninguna
MQRO_COA_WITH _FULL_DATA	Almacenamiento intermedio (Bufferlength)	Ninguna	Ninguna
MQRO_EXPIRATION _WITH_DATA	Ninguna	Ninguna	Almacenamiento intermedio (primeros 100 bytes)
MQRO_COD_WITH_DA TA	Ninguna	Almacenamiento intermedio (primeros 100 bytes)	Ninguna
MQRO_COA_WITH_DA TA	Almacenamiento intermedio (primeros 100 bytes)	Ninguna	Ninguna

PubLevel (MQLONG)

El valor inicial de este campo es 9. El nivel de suscripción de destino de esta publicación. Sólo las suscripciones con el SubLevel más alto o inferior a este valor reciben esta publicación. Este valor debe estar en el rango de cero a 9; cero es el nivel más bajo. Sin embargo, si una publicación se ha retenido, ya no está disponible para los suscriptores en niveles superiores porque se vuelve a publicar en PubLevel 1.

Para obtener información, consulte Interceptación de publicaciones.

MQPMR-Registro de colocación de mensajes

Utilice la estructura MQPMR para especificar varias propiedades de mensaje para un único destino al colocar un mensaje en una lista de distribución. MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1 .

Disponibilidad

La estructura MQPMR está disponible en las plataformas siguientes:

- AIX
- IBM i

- Linux Linux
- Solaris Solaris
- Windows Windows

y para clientes de IBM MQ conectados a estos sistemas.

Juego de caracteres y codificación

Los datos de MQPMR deben estar en el juego de caracteres que proporciona el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local que proporciona MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en la llamada MQPUT o MQPUT1 , puede especificar valores diferentes para cada cola de destino en una lista de distribución. Algunos de los campos son sólo de entrada, otros son de entrada/salida.

Nota: Esta estructura es inusual en que no tiene un diseño fijo. Los campos de esta estructura son opcionales, y la presencia o ausencia de cada campo se indica mediante los distintivos del campo *PutMsgRecFields* en MQPMO. Los campos que están presentes ***deben aparecer en el orden siguiente*** :

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Los campos ausentes no ocupan ningún espacio en el registro.

Puesto que MQPMR no tiene un diseño fijo, no se proporciona ninguna definición del mismo en los archivos de cabecera, COPY e INCLUDE para los lenguajes de programación soportados. El programador de aplicaciones debe crear una declaración que contenga los campos necesarios para la aplicación y establecer los distintivos en *PutMsgRecFields* para indicar los campos que están presentes.

Campos

No hay valores iniciales definidos para esta estructura, ya que no se proporcionan declaraciones de estructura en los archivos de cabecera, COPYy INCLUDE para los lenguajes de programación soportados. Las declaraciones de ejemplo muestran cómo declarar la estructura si se necesitan todos los campos.

Tabla 512. Campos en MQPMR

Nombre de campo	Descripción del campo
MsgId	Identificador de mensaje
CorrelId	Identificador de correlación
GroupId	Identificador de grupo
FEEDBACK	Código de comentario o razón
AccountingToken	Señal de contabilidad

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de Visual Basic para 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)

Este es el identificador de mensaje que se utilizará para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MsgId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *MsgId* . Si ese valor es MQMI_NONE, se genera un nuevo identificador de mensaje para *cada* de esos destinos (es decir, dos de esos destinos no tienen el mismo identificador de mensaje).

Si se especifica MQPMO_NEW_MSG_ID, se generan nuevos identificadores de mensaje para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa MQPMO_NEW_CORREL_ID (consulte el campo *CorrelId*).

Es un campo de entrada/salida.

CorrelId (MQBYTE24)

Es el identificador de correlación que se debe utilizar para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *CorrelId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *CorrelId* .

Si se especifica MQPMO_NEW_CORREL_ID, se genera un *único* identificador de correlación nuevo y se utiliza para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa MQPMO_NEW_MSG_ID (consulte el campo *MsgId*).

Es un campo de entrada/salida.

GroupId (MQBYTE24)

GroupId es el identificador de grupo que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *GroupId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *GroupId* . El valor se procesa tal como se documenta en Orden físico en una cola, pero con las diferencias siguientes:

- GroupId se crea a partir de QMName y una indicación de fecha y hora. Por lo tanto, para mantener un GroupId exclusivo, mantenga también los nombres de gestor de colas exclusivos. Tampoco vuelva a establecer los relojes en la máquina de gestores de colas.
- En los casos en los que se utilizaría un nuevo identificador de grupo, el gestor de colas genera un identificador de grupo diferente para cada destino (es decir, ningún destino tiene el mismo identificador de grupo).
- En los casos en los que se utilizaría el valor del campo, la llamada falla con el código de razón MQRC_GROUP_ID_ERROR

Es un campo de entrada/salida.

Feedback (MQLONG)

Este es el código de retorno que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *Feedback* en MQMD para una colocación en una sola cola.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

AccountingToken (MQBYTE32)

Es la señal de contabilidad que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *AccountingToken* en MQMD para una colocación en una sola cola. Consulte la descripción de *AccountingToken* en “MQMD - Descriptor de mensaje” en la página 421 para obtener información sobre el contenido de este campo.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

MQRFH - Cabecera de reglas y formato

La estructura MQRFH define el diseño de las reglas y la cabecera de formato. Utilice esta cabecera para enviar datos de serie en forma de pares nombre-valor.

Disponibilidad

Todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

MQFMT_RF_HEADER

Juego de caracteres y codificación

Los campos de la estructura MQRFH (incluido *NameValueString*) están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* en la estructura de cabecera que precede a la MQRFH, o por los campos de la estructura MQMD si la MQRFH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 513. Campos en MQRFH para MQRFH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQRFH_STRUC_ID	'RFH¬'
Versión (número de versión de estructura)	MQRFH_VERSION_1	1
StrucLength (longitud en bytes de la estructura MQRFH)	MQRFH_STRUC_LENGTH_FIXED	32
Codificación (codificación numérica de los datos siguientes *NameValueString*)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (especifica el identificador de juego de caracteres de los datos que siguen a *NameValueString*)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de los datos que siguen a *NameValueString*)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQRFH_NONE	0
NameValueString (serie de caracteres de longitud variable que contiene pares de nombre-valor)	Ninguno	Ninguno

Tabla 513. Campos en MQRFH para MQRFH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRFH_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRFH MyRFH = {MQRFH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura; el valor debe ser:

MQRFH_STRUC_ID
Identificador de reglas y estructura de cabecera de formato.

Para el lenguaje de programación C, también se define la constante MQRFH_STRUC_ID_ARRAY; tiene el mismo valor que MQRFH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRFH_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_1
Reglas de Version-1 y estructura de cabecera de formato.

El valor inicial de este campo es MQRFH_VERSION_1.

StrucLength (MQLONG)

Es la longitud en bytes de la estructura MQRFH, incluido el campo *NameValueString* al final de la estructura. La longitud no incluye los datos de usuario que siguen al campo *NameValueString* .

Para evitar problemas al convertir los datos de usuario en algunos entornos, *StrucLength* debe ser un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo el campo *NameValueString* :

MQRFH_STRUC_LENGTH_FIXED
Longitud de la parte fija de la estructura MQRFH.

El valor inicial de este campo es MQRFH_STRUC_LENGTH_FIXED.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a *NameValueString* ; no se aplica a los datos numéricos de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a *NameValueString* ; no se aplica a los datos de tipo carácter de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
: Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

  El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

  MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a *NameValueString*.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Se puede especificar lo siguiente:

MQRFH_NONE
: Sin distintivos.

El valor inicial de este campo es MQRFH_NONE.

Serie NameValue(MQCHARn)

Se trata de una serie de caracteres de longitud variable que contiene pares nombre-valor con el formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Cada nombre o valor debe estar separado del nombre o valor adyacente por uno o más caracteres en blanco; estos espacios en blanco no son significativos. Un nombre o valor puede contener espacios en blanco significativos añadiendo como prefijo y sufijo el nombre o valor con comillas dobles; todos los caracteres entre las comillas dobles abiertas y las comillas dobles de cierre coincidentes se tratan como significativos. En el ejemplo siguiente, el nombre es `FAMOUS_WORDS`y el valor es `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Un nombre o valor puede contener cualquier carácter que no sea el carácter nulo (que actúa como delimitador para *NameValueString*). Sin embargo, para ayudar a la interoperatividad, una aplicación puede restringir los nombres a los siguientes caracteres:

- Primer carácter: alfabético en mayúsculas o minúsculas (de la A a la Z, o de la a a la z), o subrayado.
- Caracteres subsiguientes: alfabético en mayúsculas o minúsculas, dígito decimal (de 0 a 9), subrayado, guión o punto.

Si un nombre o valor contiene una o más comillas dobles, el nombre o valor debe estar entre comillas dobles, y cada comillas dobles dentro de la serie debe duplicarse:

```
Famous_Words "The program displayed ""Hello World"""
```

Los nombres y valores distinguen entre mayúsculas y minúsculas, es decir, las letras minúsculas no se consideran iguales que las letras mayúsculas. Por ejemplo, FAMOUS_WORDS y Famous_Words son dos nombres diferentes.

La longitud en bytes de *NameValueString* es igual a *StrucLength* menos MQRFH_STRUC_LENGTH_FIXED. Para evitar problemas al convertir los datos de usuario en algunos entornos, haga que esta longitud sea un múltiplo de cuatro. Rellene *NameValueString* con espacios en blanco hasta esta longitud, o termine antes colocando un carácter nulo después del último carácter significativo de la serie. El carácter nulo y los bytes que le siguen, hasta la longitud especificada de *NameValueString*, se ignoran.

Nota: Debido a que la longitud de este campo no es fija, el campo se omite de las declaraciones de la estructura que se proporcionan para los lenguajes de programación soportados.

MQRFH2 – Reglas y formato de la cabecera 2

La cabecera MQRFH2 se basa en la cabecera MQRFH , pero permite que las series Unicode se transporten sin conversión, y puede transportar tipos de datos numéricos.La estructura MQRFH2 define el formato de las reglas version-2 y la cabecera de formato. Utilice esta cabecera para enviar datos que se han codificado utilizando una sintaxis similar a XML. Un mensaje puede contener dos o más estructuras MQRFH2 en serie, con datos de usuario siguiendo opcionalmente la última estructura MQRFH2 de la serie.

Disponibilidad

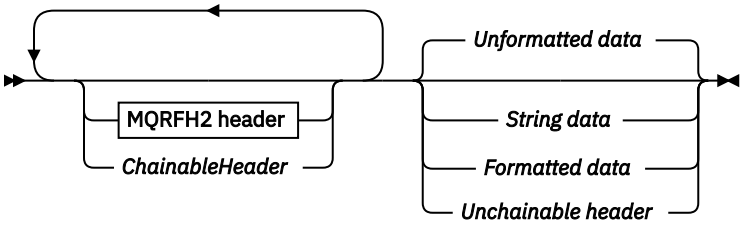
Todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

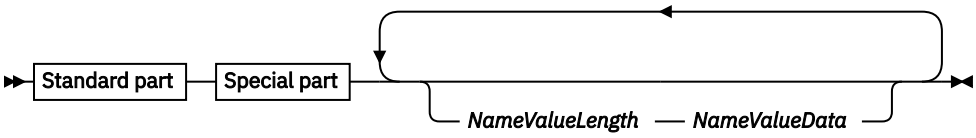
MQFMT_RF_HEADER_2

Syntax

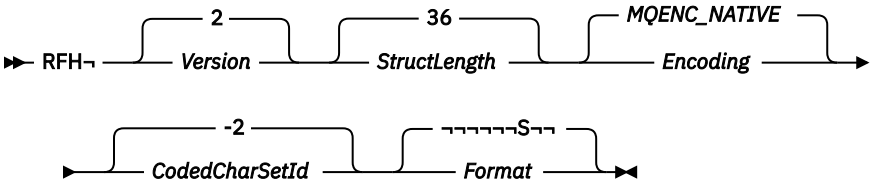
IBM MQ Message



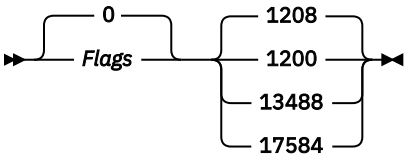
MQRFH2 header



Standard part



Special part



Juego de caracteres y codificación

Las reglas especiales se aplican al juego de caracteres y a la codificación utilizados para la estructura MQRFH2 :

- Los campos que no son *NameValueData* están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* en la estructura de cabecera que precede a MQRFH2, o por los campos de la estructura MQMD si MQRFH2 está al principio de los datos del mensaje de aplicación.

  El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

  Cuando se especifica MQGMO_CONVERT en la llamada MQGET , el gestor de colas convierte los campos MQRFH2 , distintos de *NameValueData*, en el juego de caracteres y la codificación solicitados.
- *NameValueData* está en el juego de caracteres proporcionado por el campo *NameValueCCSID* . Sólo los juegos de caracteres Unicode listados son válidos para *NameValueCCSID* ; consulte la descripción de *NameValueCCSID* para obtener más detalles.

  Algunos juegos de caracteres tienen una representación que depende de la codificación. Si *NameValueCCSID* es uno de estos juegos de caracteres, *NameValueData* debe estar en la misma codificación que los otros campos de MQRFH2.

  Cuando se especifica MQGMO_CONVERT en la llamada MQGET , el gestor de colas convierte *NameValueData* a la codificación solicitada, pero no cambia su juego de caracteres.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 514. Campos en MQRFH2 para MQRFH2

Nombre de campo	Nombre de constante	Valor de constante
StrucId (identificador de estructura)	MQRFH_STRUC_ID	'RFH¬'
Versión (número de versión de estructura)	MQRFH_VERSION_2	2

Tabla 514. Campos en MQRFH2 para MQRFH2 (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
StrucLength (longitud en bytes de la estructura MQRFH2)	MQRFH_STRUC_LENGTH_FIXED_2	36
Codificación (codificación numérica de los datos que siguen al último campo *NameValueData*)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (identificador de juego de caracteres de los datos que siguen al último campo *NameValueData*)	MQCCSI_INHERIT	-2
Formato (nombre de formato de los datos que siguen al último campo *NameValueData*)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQRFH_NONE	0
NameValueCCSID (identificador de juego de caracteres codificado de los datos en el campo *NameValueData*)	Ninguna	1208
NameValueLength (longitud en bytes de los datos en el campo *NameValueData*)	Ninguna	None
NameValueData (pares nombre-valor de propiedades de mensaje)	Ninguna	Ninguna

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRFH2_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura; el valor debe ser:

MQRFH_STRUC_ID
Identificador de reglas y estructura de cabecera de formato.

Para el lenguaje de programación C, también se define la constante MQRFH_STRUC_ID_ARRAY; tiene el mismo valor que MQRFH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRFH_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_2
Version-2 reglas y estructura de cabecera de formato.

El valor inicial de este campo es MQRFH_VERSION_2.

StrucLength (MQLONG)

Es la longitud en bytes de la estructura MQRFH2 , incluidos los campos *NameValueLength* y *NameValueData* al final de la estructura. Es válido para que haya varios pares de campos *NameValueLength* y *NameValueData* al final de la estructura, en la secuencia:

```
length1, data1, length2, data2, ...
```

StrucLength no incluye ningún dato de usuario que pueda seguir al último campo *NameValueData* al final de la estructura.

Para evitar problemas con la conversión de los datos de usuario en algunos entornos, *StrucLength* debe ser un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo los campos *NameValueLength* y *NameValueData* :

MQRFH_STRUC_LENGTH_FIXED_2
Longitud de la parte fija de la estructura MQRFH2 .

El valor inicial de este campo es MQRFH_STRUC_LENGTH_FIXED_2.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen al último campo *NameValueData* ; no se aplica a los datos numéricos de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen al último campo *NameValueData* ; no se aplica a los datos de tipo carácter de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_INHERIT.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen al último campo *NameValueData* .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor inicial de este campo es MQRFH_NONE. MQRFH_NONE debe especificarse.

MQRFH_NONE
Sin distintivos.

MQRFH_INTERNAL
La cabecera MQRFH2 contiene propiedades establecidas internamente.

MQRFH_INTERNAL es para uso del gestor de colas.

Los 16 bits principales, MQRFH_FLAGS_RESTRICTED_MASK, están reservados para los distintivos de los conjuntos de gestores de colas. Los distintivos que un usuario puede establecer se definen en los 16 bits inferiores.

NameValueCCSID (MQLONG)

Especifica el identificador de juego de caracteres codificado de los datos en el campo *NameValueData* . Es diferente del juego de caracteres de las otras series de la estructura MQRFH2 y puede ser diferente del juego de caracteres de los datos (si los hay) que siguen al último campo *NameValueData* al final de la estructura.

NameValueCCSID debe tener uno de los valores siguientes:

CCSID	Significado
1200	UTF-16, versión soportada más reciente de Unicode

CCSID	Significado
13488	UTF-16, subconjunto de Unicode versión 2.0
17584	UTF-16, subconjunto de Unicode versión 3.0 (incluye símbolo del Euro)
1208	UTF-8, versión soportada más reciente de Unicode

Para los juegos de caracteres UTF-16 , la codificación (orden de bytes) de *NameValueData* debe ser la misma que la codificación de los otros campos de la estructura MQRFH2 .

Los caracteres más allá del plano multilingüe básico Unicode (los que están por encima de U + FFFF), representados en UTF-16 por elementos de código sustitutos (X'D800'a X'DFFF'), o cuatro bytes en UTF-8, no están soportados.

Nota: Si *NameValueCCSID* no tiene uno de los valores listados anteriormente y la estructura MQRFH2 requiere conversión en la llamada MQGET, la llamada se completa con el código de razón MQRC_SOURCE_CCSID_ERROR y el mensaje se devuelve sin convertir.

El valor inicial de este campo es 1208.

NameValueLongitud (MQLONG)

La longitud del campo `NameValueData` correspondiente

Especifica la longitud en bytes de los datos del campo *NameValueData* . *NameValueLength* debe ser un múltiplo de cuatro.

Nota: Los campos *NameValueLength* y *NameValueData* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

Datos de NameValue(MQCHARn)

NameValueData es un campo de longitud variable que contiene una carpeta que contiene pares de nombre-valor de propiedades de mensaje. Una carpeta es una serie de caracteres de longitud variable que contiene datos codificados utilizando una sintaxis similar a XML. La longitud en bytes de la serie de caracteres la proporciona el campo *NameValueLength* que precede al campo *NameValueData* . La longitud debe ser un múltiplo de cuatro.

Los campos *NameValueLength* y *NameValueData* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

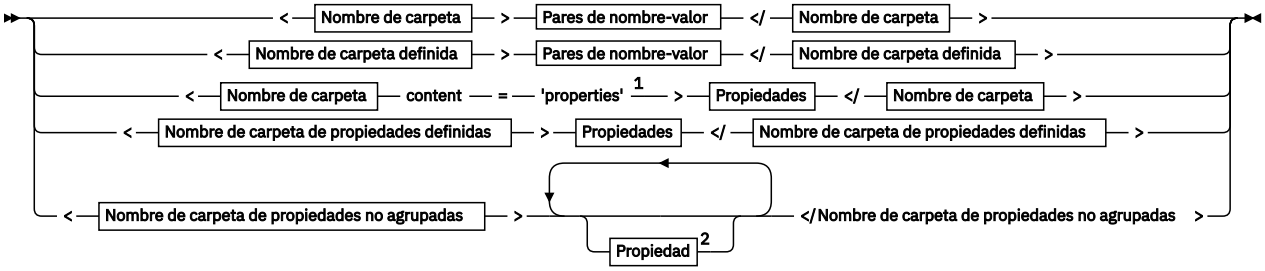
NameValueData no se convierte al juego de caracteres especificado en la llamada `MQGET` . Incluso si el mensaje se recupera con la opción `MQGMO_CONVERT` en vigor *NameValueData* permanece en su juego de caracteres original. Sin embargo, *NameValueData* se convierte a la codificación especificada en la llamada `MQGET` .

Notas:

- Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

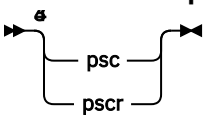
- Los términos "definidos" y "reservados" se utilizan en el diagrama de sintaxis. "Definido" significa que IBM MQutiliza el nombre. "Reservado" significa que el nombre está reservado para uso futuro por parte de IBM MQ.

NameValueData sintaxis

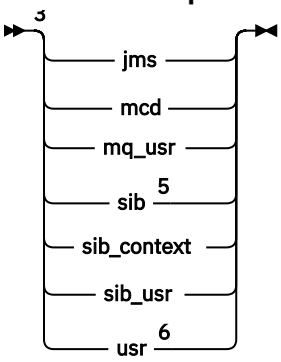


Nombre de carpeta

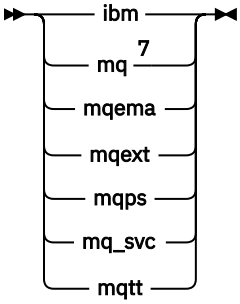
Nombre de carpeta definida



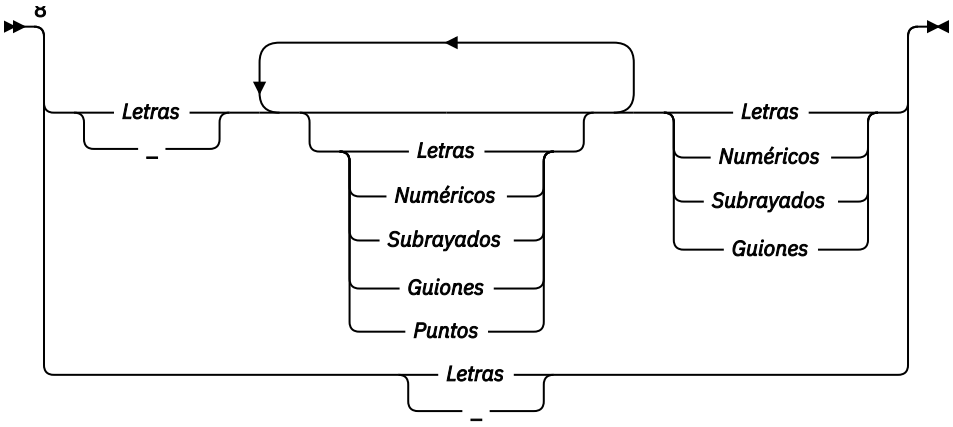
Nombre de carpeta de propiedades definidas



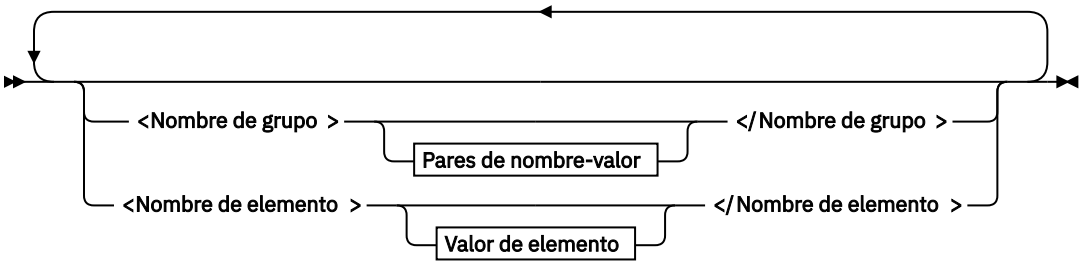
Nombre de carpeta de propiedades no agrupadas



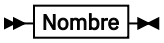
Nombre



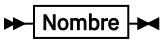
Pares de nombre-valor



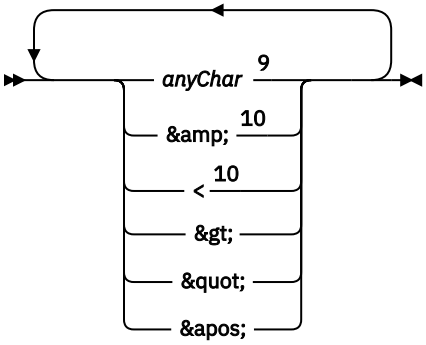
Nombre de grupo



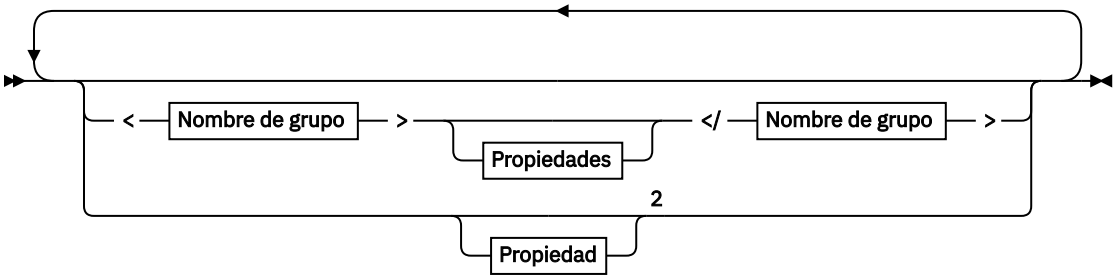
Nombre de elemento



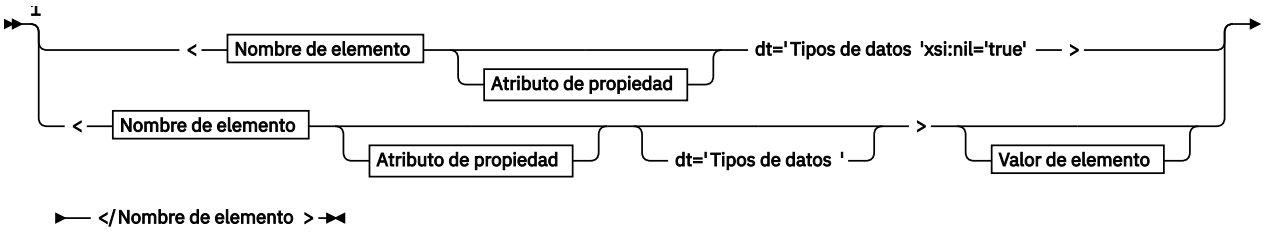
Valor de elemento



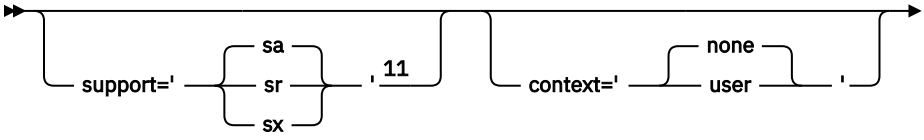
Propiedades

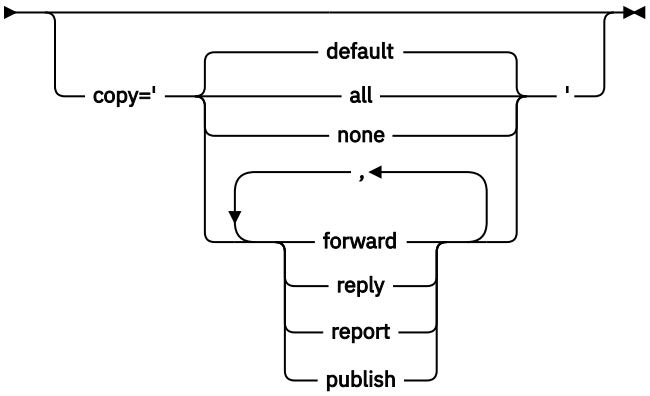


Propiedad

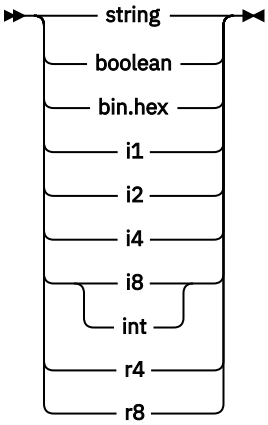


Atributo de propiedad





Tipos de datos



Notas:

[1] Las comillas dobles o las comillas simples son válidas.

[2] No utilice un nombre de propiedad no válido; consulte “Nombre de propiedad no válido” en la página 553. Utilice un nombre de propiedad reservado sólo para su finalidad definida; consulte “Nombres de propiedad definidos” en la página 553.

[3] El nombre debe estar en minúsculas.

[4] Solo se admite una carpeta `psc` y `pscr` .

[5] WebSphere Application Server Service Integration Bus ignora las carpetas sib, sib_contexty sib_usr en las cabeceras `MQRFH2` posteriores, y solo las propiedades de la primera cabecera `MQRFH2` son significativas.

[6] No debe haber más de una carpeta `usr` presente en un `MQRFH2`. Las propiedades de la carpeta `usr` no deben aparecer más de una vez.

[7] Sólo son significativas las propiedades de la primera carpeta mq . Si la carpeta es UTF-8, sólo se da soporte a caracteres UTF-8 de un solo byte. El único carácter de espacio en blanco es Unicode U+0020.
[8] Los caracteres válidos se definen en la especificación XML W3C y constan esencialmente de categorías Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, y Nd ; consulte Categorías de caracteres Unicode.
[9] Todos los caracteres son significativos. Los espacios en blanco iniciales y finales forman parte del valor del elemento.
[10] No utilice un carácter no válido; consulte "Caracteres no válidos" en la página 553. Utilice una secuencia de escape, en lugar de estos caracteres no válidos.
[11] El atributo de propiedad de soporte sólo es válido en la carpeta mq

Nombre de carpeta

NameValueData contiene una única carpeta. Para crear varias carpetas, cree varios campos *NameValueData* . Puede crear varios campos *NameValueData* en una única cabecera MQRFH2 dentro de un mensaje. De forma alternativa, puede crear varias cabeceras MQRFH2 encadenadas, cada una de las cuales contiene varios campos *NameValueData* .

El orden de las cabeceras MQRFH2 y el orden de los campos *NameValueData* no hacen ninguna diferencia en el contenido lógico de una carpeta. Si la misma carpeta está presente más de una vez en un mensaje, la carpeta se analiza como un todo. Si se produce la misma propiedad en varias instancias de la misma carpeta, se analiza como una lista.

Un análisis correcto de un MQRFH2 no se ve afectado por las formas alternativas en que una carpeta se puede almacenar físicamente en un mensaje.

Cuatro carpetas no siguen esta regla. Sólo se analiza la primera instancia de la carpeta mq, sib, sib_contexty sib_usr .

Si la misma propiedad aparece más de una vez en el contenido combinado de las cabeceras MQRFH2 encadenadas, sólo se analiza la primera instancia de la propiedad. Si una propiedad se establece utilizando una llamada de API, como por ejemplo MQSETMP, y se añade a un MQRFH2 directamente mediante una aplicación, la llamada de API tiene prioridad.

Un nombre de carpeta es el nombre de una carpeta que contiene pares de nombre-valor o grupos. Los grupos y los pares nombre-valor se pueden mezclar en el mismo nivel en el árbol de carpetas; consulte Figura 1 en la página 543. No combine un nombre de grupo y un nombre de elemento; consulte Figura 2 en la página 543

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Figura 1. Usos correctos de grupos y pares de nombre-valor

```
<group1><nvp1> value </nvp1> value </group1>
```

Figura 2. Uso incorrecto de grupos y pares nombre-valor

No utilice un nombre de carpeta no válido o reservado; consulte "Nombre de vía de acceso no válido" en la página 553 y "Carpeta reservada o nombre de carpeta de propiedades" en la página 552. Utilice un nombre de carpeta definido sólo para su finalidad definida; consulte "Nombre de carpeta definida" en la página 544.

Si añade el atributo 'content=properties' al código de nombre de carpeta, la carpeta se convierte en una carpeta de propiedades; consulte Figura 3 en la página 544.

```
<myFolder></myfolder>
<myPropertyFolder contents='properties'></myPropertyFolder>
```

Figura 3. Ejemplo de una carpeta y una carpeta de propiedades

Los nombres de carpeta distinguen entre mayúsculas y minúsculas. Los nombres de carpeta y los nombres de carpeta de propiedades comparten el mismo espacio de nombres. Deben tener nombres diferentes. `Folder1` en Figura 4 en la página 544 debe ser un nombre diferente a `Folder2` en Figura 5 en la página 544.

```
< Folder1 ><NVP1> value </NVP1></ Folder1 >
```

Figura 4. Espacio de nombres `Folder1`

```
< Folder2 content='properties'>< Property1 > value </ Property1 ></ Folder2 >
```

Figura 5. Espacio de nombres `Folder2`

Los grupos, las propiedades y los pares de nombre-valor en carpetas diferentes tienen espacios de nombres diferentes. `Property1` en Figura 5 en la página 544 es una propiedad diferente de `Property1` en Figura 6 en la página 544.

```
<Folder3 content='properties'>< Property1 > value </ Property1 ></Folder3>
```

Figura 6. Espacio de nombres `Folder3`

Las carpetas de propiedades son diferentes a las carpetas que no son de propiedades en dos aspectos importantes:

1. Las carpetas de propiedades contienen propiedades y las carpetas que no son de propiedades contienen pares de nombre-valor. Las carpetas difieren ligeramente, sintácticamente.
2. Utilice las interfaces definidas, como las propiedades `MQI` o las propiedades de mensaje `JMS` , para acceder a las propiedades de mensaje. Las interfaces garantizan que las carpetas de propiedades de `MQRFH2` estén bien formadas. Una carpeta de propiedades bien formada es interoperable entre gestores de colas en distintas plataformas y distintos releases.

La propiedad de mensaje `MQI` es una forma sólida de leer y escribir un `MQRFH2`, y evita las dificultades de analizar un `MQRFH2` correctamente.

Nombre de carpeta definida

Un nombre de carpeta definido es el nombre de una carpeta que está reservada para que la utilice IBM MQu otro producto. No cree una carpeta con el mismo nombre y no añada sus propios pares nombre-valor a las carpetas. Las carpetas definidas son `psc` y `pscr`.

La publicación/suscripción en cola utiliza `psc` y `pscr` .

Un mensaje segmentado colocado con `MQMF_SEGMENT` o `MQMF_SEGMENTATION_ALLOWED` no puede contener un `MQRFH2` con un nombre de carpeta definido. El `MQPUT` falla con el código de razón 2443, `MQRC_SEGMENTATION_NOT_ALLOWED`.

Nombre de carpeta de propiedades definidas

Un nombre de carpeta de propiedades definido es el nombre de una carpeta de propiedades utilizada por IBM MQu otro producto. Para ver los nombres de las carpetas y su contenido, consulte Carpetas de propiedades. Los nombres de carpeta de propiedades definidas son un subconjunto de todos los nombres de carpeta reservados por IBM MQ; consulte "Carpeta reservada o nombre de carpeta de propiedades" en la página 552.

Cualquier elemento almacenado en una carpeta de propiedades definida es una propiedad. Un elemento almacenado en una carpeta de propiedades definida no debe tener un atributo `content='properties'` .

Solo puede añadir propiedades a las carpetas de propiedades definidas `usr`, `mq_usr`y `sib_usr`. En otras carpetas de propiedades, como `mq` y `sib`, IBM MQ ignora o descarta las propiedades que no reconoce.

La descripción de cada carpeta de propiedades definida lista las propiedades que IBM MQ ha definido que pueden utilizar los programas de aplicación. A algunas de las propiedades se accede indirectamente estableciendo u obteniendo una propiedad JMS , y a otras se accede directamente utilizando las llamadas MQI `MQSETMP` y `MQINQMP` .

Las carpetas de propiedades definidas también contienen otras propiedades que IBM MQ ha reservado, pero a las que las aplicaciones no tienen acceso. Los nombres de las propiedades reservadas no se listan. No hay propiedades reservadas presentes en las carpetas de propiedades `usr`, `mq_usr`y `sib_usr` . Pero no cree propiedades con nombres de propiedad no válidos; consulte "Nombre de propiedad no válido" en la página 553.

Carpetas de propiedades

jms

`jms` contiene campos de cabecera JMS y propiedades JMSX que no se pueden expresar completamente en `MQMD`. La carpeta `jms` siempre está presente en un JMS `MQRFH2`.

Tabla 515. nombre de propiedad jms, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSDestination	jms.Dst	string	<jms><Dst> *destination* </Dst></jms>
JMSExpiration	jms.Exp	i8	<jms><Exp> *expiration* </Exp></jms>
Correlación JMS	jms.Cid	string	<jms><Cid> *correlationId* </Cid></jms>
JMSDelivery	jms.Dlv	i4	<jms><Dlv> *delivery* </Dlv></jms>
JMSPriority	jms.Pri	i4	<jms><Pri> *priority* </Pri></jms>
JMSReplyTo	jms.Rto	string	<jms><Rto> *replyToURI* </Rto></jms>
JMSTimeStamp	jms.Tms	i8	<jms><Tms> *timestamp* </Tms></jms>
JMSXGroupID	jms.Gid	string	<jms><Gid> *groupId* </Gid></jms>
JMSXGroupSeq	jms.Seq	i4	<jms><Seq> *messageSequenceNo* </Seq></jms>

No añada sus propias propiedades en la carpeta jms.

mcd

mcd contiene propiedades que describen el formato del mensaje. Por ejemplo, la propiedad de dominio de servicio de mensajes Msd identifica un mensaje JMS como JMSTextMessage, JMSBytesMessage, JMSStreamMessage, JMSMapMessage, JMSObjectMessage o nulo.

La carpeta mcd siempre está presente en un mensaje JMS que contiene un MQRFH2.

Siempre está presente en un mensaje que contiene un MQRFH2 enviado desde IBM Integration Bus. Describe el dominio, formato, tipo y conjunto de mensajes de un mensaje.

Tabla 516. mcd nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	mcd.Msd	string	<mcd><Msd>*messageDomain*</Msd></mcd>
	mcd.Set	string	<mcd><Set>*messageDomain*</Set></mcd>
	mcd.Type	string	<mcd><Type>*messageDomain*</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>*messageDomain*</Fmt></mcd>

No añada sus propias propiedades en la carpeta mcd.

mq_usr

mq_usr contiene propiedades definidas por la aplicación que no están expuestas como propiedades definidas por el usuario de JMS . Las propiedades que no cumplen los requisitos de JMS se pueden colocar en esta carpeta.

Puede crear propiedades en la carpeta mq_usr . Las propiedades que crea en mq_usr son como las propiedades que crea en carpetas nuevas con el atributo content='properties' .

sib

sib contiene propiedades de mensaje del sistema del bus de integración de servicios WebSphere Application Server (WAS/SIB). Las propiedades sib no se exponen como propiedades JMS en aplicaciones IBM MQ JMS porque no son de los tipos soportados. Por ejemplo, algunas propiedades sib no se pueden exponer como propiedades JMS porque son matrices de bytes. Algunas propiedades sib se exponen a las aplicaciones WAS/SIB como propiedades JMS_IBM_* ; estas incluyen las propiedades de vías de acceso de direccionamiento inverso y de reenvío.

No añada sus propias propiedades en la carpeta sib.

sib_context

sib_context contiene propiedades de mensaje del sistema WAS/SIB que no están expuestas a aplicaciones de usuario WAS/SIB o como propiedades JMS . sib_context contiene propiedades de seguridad y transaccionales que se utilizan para los servicios web.

No añada sus propias propiedades en la carpeta sib_context.

sib_usr

sib_usr contiene propiedades de mensaje de usuario WAS/SIB que no se exponen como propiedades de usuario JMS porque no son de tipos soportados. sib_usr está expuesto a aplicaciones WAS/SIB en la interfaz SIMessage ; consulte Desarrollo de integración de servicios.

El tipo de una propiedad `sib_usr` debe ser `bin.hex`y el valor debe estar en el formato correcto. Si una aplicación IBM MQ escribe un elemento con tipo `bin.hex` en la carpeta con un formato incorrecto, la aplicación recibe un `IOException`. Si el tipo de datos de la propiedad no es `bin.hex` , la aplicación recibe un `ClassCastException`.

No intente que las propiedades de usuario de JMS estén disponibles para WAS/SIB utilizando esta carpeta; en su lugar, utilice la carpeta `usr` .

Puede crear propiedades en la carpeta `sib_usr` .

usr

`usr` contiene propiedades JMS definidas por la aplicación asociadas al mensaje. La carpeta `usr` solo está presente si una aplicación ha establecido una propiedad definida por la aplicación.

`usr` es la carpeta de propiedades predeterminada. Si una propiedad se establece sin un nombre de carpeta, se coloca en la carpeta `usr` .

Tabla 517. nombre de propiedad usr, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	`usr.contentType`	string	`<usr><contentType>text/xml; charset=utf-8</contentType></usr>`
	`usr.endPointURL`	string	`<usr><endPointURL>` *URI* `</endPointURL></usr>`
	`usr.targetService`	string	`<usr><targetService>` *serviceName* `</targetService></usr>`
	`usr.soapAction`	string	`<usr><soapAction>` *name* `</soapAction></usr>`
	`usr.transportVersion`	string	`<usr><transportVersion>` *version* `</transportVersion></usr>`

Puede crear propiedades en la carpeta `usr` .

Un mensaje segmentado colocado con `MQMF_SEGMENT` o `MQMF_SEGMENTATION_ALLOWED` no puede contener un `MQRFH2` con un nombre de carpeta de propiedades definido. El `MQPUT` falla con el código de razón 2443, `MQRC_SEGMENTATION_NOT_ALLOWED`.

Nombre de carpeta de propiedades no agrupadas

ibm

`ibm` contiene propiedades que sólo utiliza IBM MQ.

Tabla 518. `ibm` nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	`ibm.rfp`	string	`<ibm><rfp>fingerprint</rfp></ibm>`

No añada sus propias propiedades en la carpeta `ibm`.

mq

mq contiene propiedades que sólo utiliza IBM MQ.

Las restricciones siguientes se aplican a las propiedades de la carpeta mq :

- MQsólo actúa sobre las propiedades de la primera carpeta mq significativa del mensaje; las propiedades de cualquier otra carpeta mq del mensaje se ignoran.
- Sólo se permiten caracteres UTF-8 de un solo byte en la carpeta. Un carácter de varios bytes en la carpeta puede hacer que falle el análisis y que se rechace el mensaje.
- No utilice series de escape en la carpeta. Una serie de escape se trata como el valor real del elemento.
- Sólo el carácter Unicode U+0020 se trata como un espacio en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden hacer que falle el análisis de la carpeta y que se rechace el mensaje.

Si el análisis de la carpeta mq falla, o si la carpeta no observa estas restricciones, el mensaje se rechaza con el código de razón 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR.

No añada sus propias propiedades en la carpeta mq.

mqema

mqema contiene propiedades que sólo utiliza WebSphere Application Server. La carpeta se ha sustituido por mqext.

No añada sus propias propiedades en la carpeta mqema.

mqext

mqext contiene los siguientes tipos de propiedad:

- Propiedades que únicamente utiliza WebSphere Application Server.
- Propiedades relacionadas con la entrega retardada de mensajes.

La carpeta está presente si la aplicación ha establecido como mínimo una de las propiedades definidas por IBM o ha utilizado el retardo el retardo de entrega.

Tabla 519. mqext nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
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>

No añada sus propias propiedades en la carpeta mqext.

mqps

mqps contiene propiedades que solo son utilizadas por la publicación/suscripción de IBM MQ. La carpeta sólo está presente si la aplicación ha establecido al menos una de las propiedades de publicación/suscripción integradas.

Tabla 520. mqps nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
MQTopicString	mqps.Top	string	<mqps><Top>*topicString*</Top></mqps>
MQSubUserData	mqps.Sud	string	<mqps><Sud>*subscriberUserData*...</Sud></mqps>
MQIsRetained	mqps.Ret	boolean	<mqps><Ret>*isRetained*</Ret></mqps>
MQPubOptions	mqps.Pub	i8	<mqps><Pub>*publicationOptions*</Pub></mqps>
MQPubLevel	mqps.Pbl	i8	<mqps><Pbl>*publicationLevel*</Pbl></mqps>
MQPubTime	mqpse.Pts	string	<mqps><Pts>*publicationTime*</Pts></mqps>
MQPubSeqNum	mqpse.Seq	i8	<mqps><Seq>*publicationSequenceNumber*</Seq></mqps>
MQPubStrIntData	mqpse.Sid	string	<mqps><Sid>*publicationData*</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>*messageFormat*</Pfmt></mqps>

No añada sus propias propiedades en la carpeta mqps.

mq_svc

mq_svc contiene propiedades utilizadas por SupportPac MA93.

No añada sus propias propiedades en la carpeta mq_svc.

mqtt

mqtt contiene propiedades utilizadas por MQ Telemetry

Tabla 521. nombre de propiedad mqtt, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
	mqtt.clientId	string	<mqtt><clientId> *topicString* </clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos> *qualityOfService* </qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid> *messageIdentifier* </msgid></mqtt>

No añada sus propias propiedades en la carpeta mqtt.

Un mensaje segmentado colocado con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED no puede contener un MQRFH2 con un nombre de carpeta de propiedades no agrupado. El MQPUT falla con el código de razón 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Pares de nombre-valor

En el diagrama de sintaxis, "Pares de nombre-valor" describe el contenido de una carpeta ordinaria. Una carpeta ordinaria contiene grupos y elementos. Un elemento es un par nombre-valor. Un grupo contiene elementos y otros grupos.

En términos de árboles, los elementos son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser un nodo interno y un nodo de hoja al mismo tiempo; consulte Figura 2 en la página 543.

Propiedades

En el diagrama de sintaxis, "Propiedades" describe el contenido de una carpeta de propiedades. Una carpeta de propiedades contiene grupos y propiedades. Una propiedad es un par nombre-valor con un atributo de tipo de datos opcional. Un grupo contiene propiedades y otros grupos.

En términos de árboles, las propiedades son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta de propiedades, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser un nodo interno y un nodo de hoja al mismo tiempo; consulte Figura 2 en la página 543.

Propiedad

Una propiedad de mensaje es un par nombre-valor en una carpeta de propiedades. Opcionalmente, puede incluir un atributo de tipo de datos y un atributo de propiedad; para ver un ejemplo, consulte el código siguiente. Si se omite el atributo de tipo de datos, el tipo de propiedad es `string`.

```
<pf><p1 dt='i8' > value </p1></pf>
```

El nombre de una propiedad de mensaje es su nombre completo de vía de acceso, con la sintaxis <> de tipo XML, sustituida por puntos. Por ejemplo, `myPropertyFolder1.myGroup1.myGroup2.myProperty1` se correlaciona con una serie *`NameValueData`* como se indica a continuación. La serie se formatea para facilitar la lectura.

```
<myPropertyFolder1>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
        </myGroup2>
    </myGroup1>
</myPropertyFolder1>
```

Una carpeta de propiedades puede contener varias propiedades. Por ejemplo, las propiedades en Figura 7 en la página 550 se correlacionan con la carpeta de propiedades en Figura 8 en la página 550

```
 myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Figura 7. Varias propiedades con el mismo nombre raíz

```
<myPropertyFolder1>
    <myProperty4>value</myProperty4>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
            <myProperty2>value</myProperty2>
        </myGroup2>
        <myProperty3>value</myProperty3>
    </myGroup1>
</myPropertyFolder1>
```

Figura 8. Correlación de varios nombres de propiedad

Nombre

Un nombre debe empezar por una *Carta* o un *Subrayado*. No debe contener un *Colón*, no debe terminar en un *Periodo* y contener solo *Cartas*, *Numerales*, *Subrayados*, *guiones*y *Puntos*. Los caracteres válidos se definen en la especificación XML W3C y constan esencialmente de categorías Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` y `Nd` ; consulte Categorías de caracteres Unicode.

La vía de acceso completa de una propiedad o par nombre-valor no debe romper la regla descrita en "Nombre de vía de acceso no válido" en la página 553. Las vías de acceso están restringidas a 4095 bytes, no deben contener caracteres de compatibilidad Unicode y no deben empezar por la serie `XML`.

Nombre de grupo

Un nombre de grupo tiene la misma sintaxis que un nombre. Los nombres de grupo son opcionales. Las propiedades y los pares nombre-valor se pueden colocar en la raíz de una carpeta. Utilice grupos si ayuda a organizar las propiedades y los pares nombre-valor.

Nombre de elemento

Un nombre de elemento tiene la misma sintaxis que un nombre.

Valor de elemento

Un valor de elemento incluye todos los espacios en blanco entre el código `< Element name >` y el `< /Element name >`. No utilice los dos caracteres < y & en un valor. Sustituya entonces por < y `&`.

Atributos de Property

Los campos de descriptor de propiedad de correlación de atributos de propiedad: las correlaciones son las siguientes:

Soporte

sa (valor predeterminado)
`MQPD_SUPPORT_OPTIONAL`

sr
`MQPD_SUPPORT_REQUIRED`

sx
`MQPD_SUPPORT_REQUIRED_IF_LOCAL`

Contexto

none (valor predeterminado)
`MQPD_NO_CONTEXT`

user
`MQPD_USER _CONTEXT`

CopyOptions

hacia adelante
`MQPD_COPY_FORWARD`

responder
`MQPD_COPY_REPLY`

informe
`MQPD_COPY_REPORT`

publicar
`MQPD_COPY_PUBLISH`

todos
`MQPD_COPY_ALL`

No utilice `all` en combinación con otras opciones.

valor predeterminado
`MQPD_COPY_DEFAULT`

No utilice `default` en combinación con otras opciones. `default` es el mismo que `forward` + `report` + `publish`.

Ninguno
`MQPD_COPY_NONE`

No utilice none en combinación con otras opciones.

Los atributos de la propiedad Soporte sólo son aplicables a las propiedades de la carpeta `mq` .

Los atributos de propiedad Contexto y CopyOptions son aplicables a todas las carpetas de propiedades.

Tipos de datos

Los tipos de datos de `MQRFH2` se correlacionan con los tipos de propiedad de mensaje como se indica a continuación:

Tabla 522. Correlaciones de tipos de datos

Tipo de datos de `MQRFH2`	Tipo de propiedad de mensaje
`bin.hex`	`MQBYTE[]`
`boolean`	`MQBOOL`
`i1`	`MQINT8`
`i2`	`MQINT16`
`i4`	`MQINT32`
`i8`	`MQINT64`
`r4`	`MQFLOAT32`
`r8`	`MQFLOAT64`
`string`	`MQCHAR[]`

Se presupone que cualquier elemento sin un tipo de datos es del tipo `string`.

Un valor nulo se indica mediante el atributo de elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos. Por ejemplo, la propiedad siguiente tiene un valor nulo:

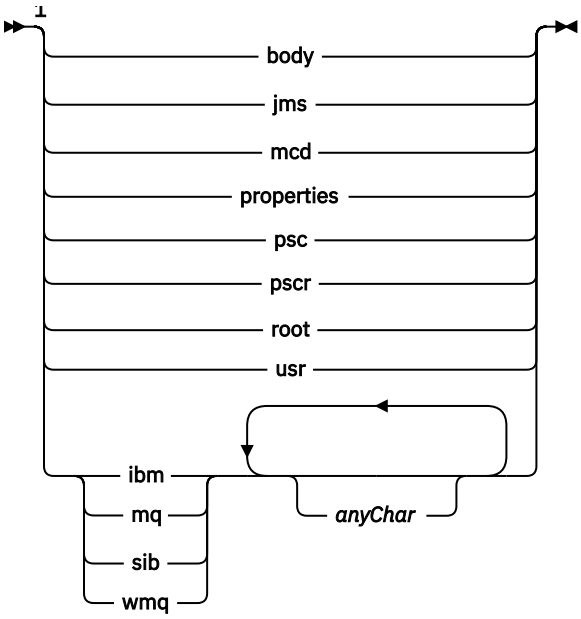
```
<NullProperty
xsi:nil='true'></NullProperty>
```

Una propiedad de serie de bytes o caracteres puede tener un valor vacío. Un valor vacío se representa mediante un elemento `MQRFH2` con un valor de elemento de longitud cero. Por ejemplo, la propiedad siguiente tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Carpeta reservada o nombre de carpeta de propiedades

Restrinja el nombre de una carpeta o carpeta de propiedades para que no empiece con ninguna de las series siguientes. Los prefijos están reservados para los nombres de carpeta o propiedad creados por IBM.

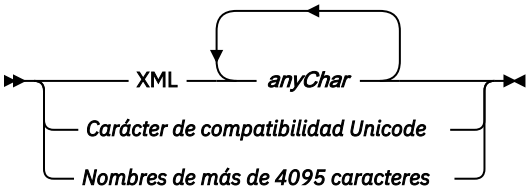


Notas:

[1] Un nombre de propiedad o carpeta reservado contiene cualquier combinación de letras minúsculas y mayúsculas.

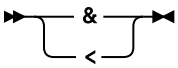
Nombre de vía de acceso no válido

Restrinja la vía de acceso completa de un par nombre-valor o una propiedad para no incluir ninguna de las series siguientes.



Caracteres no válidos

Utilice siempre las secuencias de escape `&` y < en lugar de los literales "&" y "<".

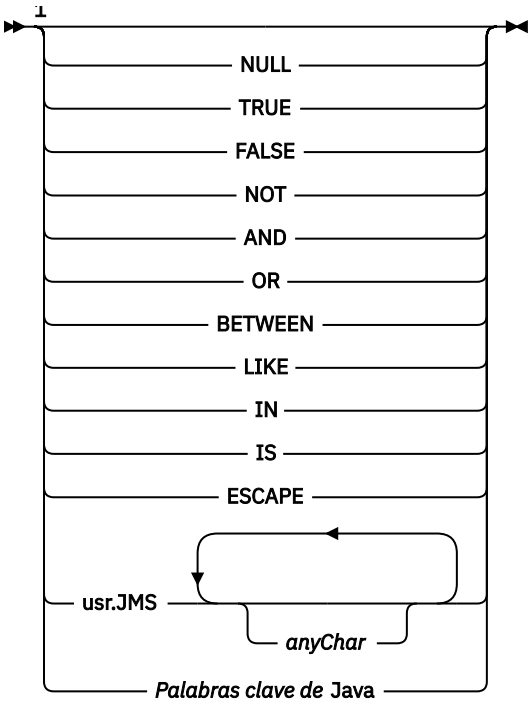


Nombres de propiedad definidos

Los nombres de propiedad definidos son los nombres de las propiedades definidas por IBM MQ, u otros productos, y utilizadas por IBM MQ y las aplicaciones de usuario. Las propiedades definidas sólo existen en las carpetas de propiedades definidas. Los nombres de propiedad definidos se describen en la descripción de las carpetas de propiedades; consulte Carpetas de propiedades.

Nombre de propiedad no válido

No construya nombres de propiedad que coincidan con la regla siguiente. La regla se aplica a la vía de acceso de propiedad completa que nombra una propiedad y no sólo al nombre del elemento de propiedad.



Notas:

[1] Un nombre de propiedad no válido puede contener cualquier combinación de mayúsculas y minúsculas.

Atributos no válidos

Las carpetas de propiedades y las propiedades sólo pueden incluir “Atributos de Property” en la página 551 y “Tipos de datos” en la página 552soportados.

Los atributos de tipo XML no soportados, por ejemplo, los nombres con valores de serie entrecomillados, que se incluyen en carpetas de propiedades o propiedades pueden eliminarse.

Atributos de tipo XML incluidos en carpetas no de propiedad o elementos no de propiedad que permanecen en cabeceras MQRFH2 .

MQRMH - Cabecera de mensaje de referencia

La estructura MQRMH define el formato de una cabecera de mensaje de referencia. Esta cabecera se utiliza con salidas de canal de mensajes escritas por el usuario para enviar cantidades extremadamente grandes de datos (denominadas *datos masivos*) de un gestor de colas a otro. La diferencia en comparación con la mensajería normal es que los datos masivos no se almacenan en una cola; en su lugar, sólo se almacena en la cola una *referencia* a los datos masivos. Esto reduce la posibilidad de que los recursos de IBM MQ se agoten en un pequeño número de mensajes extremadamente grandes.

Disponibilidad

La estructura MQRMH está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

Nombre de formato

MQFMT_REF_MSG_HEADER

Juego de caracteres y codificación

Los datos de tipo carácter en MQRMH, y las series a las que se dirigen los campos de desplazamiento, deben estar en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId** . Los datos numéricos en MQRMH deben estar en la codificación de máquina nativa; esto viene dado por el valor de MQENC_NATIVE para el lenguaje de programación C.

Establezca el juego de caracteres y la codificación de MQRMH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQRMH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQRMH (todos los demás casos).

Utilización

Una aplicación coloca un mensaje que consta de una MQRMH, pero omitiendo los datos masivos. Cuando un agente de canal de mensajes (MCA) lee el mensaje de la cola de transmisión, se invoca una salida de mensaje proporcionada por el usuario para procesar la cabecera de mensaje de referencia. La salida puede añadir al mensaje de referencia los datos masivos identificados por la estructura MQRMH, antes de que el MCA envíe el mensaje a través del canal al siguiente gestor de colas.

En el extremo receptor, debe existir una salida de mensajes que espere mensajes de referencia. Cuando se recibe un mensaje de referencia, la salida debe crear el objeto a partir de los datos masivos que siguen a la MQRMH en el mensaje y, a continuación, pasar el mensaje de referencia sin los datos masivos. El mensaje de referencia puede ser recuperado posteriormente por una aplicación que lee el mensaje de referencia (sin los datos masivos) de una cola.

Normalmente, la estructura MQRMH es todo lo que hay en el mensaje. Sin embargo, si el mensaje está en una cola de transmisión, una o más cabeceras adicionales preceden a la estructura MQRMH.

También se puede enviar un mensaje de referencia a una lista de distribución. En este caso, la estructura MQDH y sus registros relacionados preceden a la estructura MQRMH cuando el mensaje está en una cola de transmisión.

Nota: No envíe un mensaje de referencia como un mensaje segmentado, porque la salida de mensaje no puede procesarlo correctamente.

Conversión de datos

A efectos de conversión de datos, la conversión de la estructura MQRMH incluye la conversión de los datos del entorno de origen, el nombre del objeto de origen, los datos del entorno de destino y el nombre del objeto de destino. Cualquier otro byte dentro de *StrucLength* bytes del inicio de la estructura se descarta o tiene valores no definidos después de la conversión de datos. Los datos masivos se convierten siempre que todas las sentencias siguientes sean verdaderas:

- Los datos masivos están presentes en el mensaje cuando se realiza la conversión de datos.
- El campo *Format* en MQRMH tiene un valor distinto de MQFMT_NONE.
- Existe una salida de conversión de datos escrita por el usuario con el nombre de formato especificado.

Sin embargo, tenga en cuenta que normalmente los datos masivos no están presentes en el mensaje cuando el mensaje está en una cola y que, como resultado, los datos masivos se convierten mediante la opción MQGMO_CONVERT.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 523. Campos en MQRMH para MQRMH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQRMH_STRUC_ID	'RMH¬'
Versión (número de versión de estructura)	MQRMH_VERSION_1	1
StrucLength (longitud total de MQRMH, incluidas las series al final de los campos fijos, pero no los datos masivos)	Ninguna	0
Codificación (codificación numérica de datos masivos)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (identificador de juego de caracteres de datos masivos)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de datos masivos)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos de mensaje de referencia)	MQRMHF_NOT_LAST	0
ObjectType (tipo de objeto)	Ninguna	Espacios en blanco
ObjectInstanceId (identificador de instancia de objeto)	MQOII_NONE	Nulos
SrcEnvLength (longitud de datos de entorno de origen)	Ninguna	0
SrcEnvDesplazamiento (desplazamiento de datos de entorno de origen)	Ninguna	0
SrcNameLongitud (longitud del nombre de objeto de origen)	Ninguna	0
SrcNameDesplazamiento (desplazamiento del nombre de objeto de origen)	Ninguna	0
DestEnvLength (longitud de los datos de entorno de destino)	Ninguna	0
DestEnvDesplazamiento (desplazamiento de datos de entorno de destino)	Ninguna	0
DestNameLongitud (longitud del nombre de objeto de destino)	Ninguna	0
DestNameDesplazamiento (desplazamiento del nombre de objeto de destino)	Ninguna	0
DataLogicalLength (longitud de datos masivos)	Ninguna	0
DataLogical (desplazamiento bajo de datos masivos)	Ninguna	0
DataLogicalOffset2 (desplazamiento alto de datos masivos)	Ninguna	0

Tabla 523. Campos en MQRMH para MQRMH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRMH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Este es el identificador de estructura; el valor debe ser:

MQRMH_STRUC_ID
Identificador de la estructura de cabecera de mensaje de referencia.

Para el lenguaje de programación C, la constante MQRMH_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQRMH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRMH_STRUC_ID.

Versión (MQLONG)

El número de versión de la estructura. El valor debe ser:

MQRMH_VERSION_1
Estructura de cabecera de mensaje de referencia Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQRMH_CURRENT_VERSION
Versión actual de la estructura de cabecera de mensaje de referencia.

El valor inicial de este campo es MQRMH_VERSION_1.

StrucLength (MQLONG)

La longitud total de MQRMH, incluidas las series al final de los campos fijos, pero no los datos masivos.

El valor inicial de este campo es cero.

Encoding (MQLONG)

Especifica la codificación numérica de los datos masivos; no se aplica a los datos numéricos de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos masivos; no se aplica a los datos de tipo carácter de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres de los datos que siguen a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del campo `PutApplType` en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre de formato de los datos masivos.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *`Format`* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Estos son distintivos de mensaje de referencia. Se definen los distintivos siguientes:

MQRMHF_LAST
Este distintivo indica que el mensaje de referencia representa o contiene la última parte del objeto referenciado.

MQRMHF_NOT_LAST
El mensaje de referencia no contiene ni representa la última parte del objeto. MQRMHF_NOT_LAST ayuda a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial de este campo es MQRMHF_NOT_LAST.

ObjectType (MQCHAR8)

Es un nombre que la salida de mensajes puede utilizar para reconocer los tipos de mensajes de referencia a los que da soporte. El nombre debe ajustarse a las mismas reglas que el campo *Format* , consulte "Format (MQCHAR8)" en la página 560.

El valor inicial de este campo es de 8 blancos.

ObjectInstanceId (MQBYTE24)

Utilice este campo para identificar una instancia específica de un objeto. Si no es necesario, establézcalo en el valor siguiente:

MQOII_NONE
No se ha especificado ningún identificador de instancia de objeto. El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQOII_NONE_ARRAY; tiene el mismo valor que MQOII_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQOII_NONE.

SrcEnvLongitud (MQLONG)

La longitud de los datos de entorno de origen. Si este campo es cero, no hay datos de entorno de origen y se ignora *SrcEnvOffset* .

El valor inicial de este campo es 0.

SrcEnvDesplazamiento (MQLONG)

Este campo especifica el desplazamiento de los datos de entorno de origen desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos de entorno de origen, si el creador los conoce. Por ejemplo, en Windows los datos de entorno de origen pueden ser la vía de acceso del directorio del objeto que contiene los datos masivos. Sin embargo, si el creador no conoce los datos de entorno de origen, la salida de mensaje proporcionada por el usuario debe determinar la información de entorno necesaria.

La longitud de los datos de entorno de origen la proporciona *SrcEnvLength* ; si esta longitud es cero, no hay datos de entorno de origen y se ignora *SrcEnvOffset* . Si está presente, los datos de entorno de origen deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos de entorno se inician inmediatamente después del último campo fijo de la estructura o que son contiguos con cualquiera de los datos direccionados por los campos *SrcNameOffset*, *DestEnvOffset*y *DestNameOffset* .

El valor inicial de este campo es 0.

SrcNameLongitud (MQLONG)

Longitud del nombre de objeto de origen. Si este campo es cero, no hay ningún nombre de objeto de origen y se ignora *SrcNameOffset* .

El valor inicial de este campo es 0.

SrcNameDesplazamiento (MQLONG)

Este campo especifica el desplazamiento del nombre de objeto de origen desde el inicio de la estructura MQRMH. El nombre de objeto de origen puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de origen, la salida de mensaje proporcionada por el usuario debe identificar el objeto al que se debe acceder.

La longitud del nombre de objeto de origen viene dada por *SrcNameLength* ; si esta longitud es cero, no hay ningún nombre de objeto de origen y se ignora *SrcNameOffset* . Si está presente, el nombre de objeto de origen debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de origen es contiguo a ninguno de los datos a los que se dirigen los campos *SrcEnvOffset*, *DestEnvOffset*y *DestNameOffset* .

El valor inicial de este campo es 0.

DestEnvLongitud (MQLONG)

Es la longitud de los datos de entorno de destino. Si este campo es cero, no hay datos de entorno de destino y se ignora *DestEnvOffset* .

DestEnvDesplazamiento (MQLONG)

Este campo especifica el desplazamiento de los datos de entorno de destino desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos del entorno de destino, si el creador los conoce. Por ejemplo, en Windows los datos de entorno de destino pueden ser la vía de acceso del directorio del objeto donde se van a almacenar los datos masivos. Sin embargo, si el creador no conoce los datos de entorno de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar la información de entorno necesaria.

La longitud de los datos de entorno de destino la proporciona *DestEnvLength* ; si esta longitud es cero, no hay datos de entorno de destino y se ignora *DestEnvOffset* . Si está presente, los datos de entorno de destino deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos del entorno de destino son contiguos a ninguno de los datos direccionado por los campos *SrcEnvOffset*, *SrcNameOffset*y *DestNameOffset* .

El valor inicial de este campo es 0.

DestNameLongitud (MQLONG)

Longitud del nombre de objeto de destino. Si este campo es cero, no hay ningún nombre de objeto de destino y se ignora *DestNameOffset* .

DestNameDesplazamiento (MQLONG)

Este campo especifica el desplazamiento del nombre de objeto de destino desde el inicio de la estructura MQRMH. El nombre de objeto de destino puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario identificar el objeto que se va a crear o modificar.

La longitud del nombre de objeto de destino viene dada por *DestNameLength* ; Si esta longitud es cero, no hay ningún nombre de objeto de destino y se ignora *DestNameOffset* . Si está presente, el nombre de objeto de destino debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de destino es contiguo a ninguno de los datos a los que se dirigen los campos *SrcEnvOffset*, *SrcNameOffset*y *DestEnvOffset* .

El valor inicial de este campo es 0.

Longitud DataLogical(MQLONG)

El campo *DataLogicalLength* especifica la longitud de los datos masivos a los que hace referencia la estructura MQRMH.

Si los datos masivos están realmente presentes en el mensaje, los datos empiezan en un desplazamiento de *StrucLength* bytes desde el inicio de la estructura MQRMH. La longitud de todo el mensaje menos *StrucLength* proporciona la longitud de los datos masivos presentes.

Si los datos están presentes en el mensaje, *DataLogicalLength* especifica la cantidad de datos que son relevantes. El caso normal es que *DataLogicalLength* tenga el mismo valor que la longitud de los datos presentes en el mensaje.

Si la estructura MQRMH representa los datos restantes en el objeto (empezando por el desplazamiento lógico especificado), puede utilizar el valor cero para *DataLogicalLength*, siempre que los datos masivos no estén realmente presentes en el mensaje.

Si no hay datos presentes, el final de MQRMH coincide con el final del mensaje.

El valor inicial de este campo es 0.

Desplazamiento DataLogical(MQLONG)

Este campo especifica el desplazamiento bajo de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. El desplazamiento de los datos masivos desde el inicio del objeto se denomina *desplazamiento lógico*. Este no es el desplazamiento físico de los datos masivos desde el inicio de la estructura MQRMH; dicho desplazamiento lo proporciona *StrucLength*.

Para permitir que se envíen objetos grandes utilizando mensajes de referencia, el desplazamiento lógico se divide en dos campos, y el desplazamiento lógico real viene dado por la suma de estos dos campos:

- *DataLogicalOffset* representa el resto obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es un valor comprendido entre 0 y 999.999.999.
- *DataLogicalOffset2* representa el resultado obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es el número de múltiplos completos de 1 000 000 000 000 que existen en el desplazamiento lógico. El número de múltiplos está en el rango de 0 a 999.999.999.

El valor inicial de este campo es 0.

DataLogicalOffset2 (MQLONG)

Este campo especifica el desplazamiento alto de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. Es un valor comprendido entre 0 y 999.999.999. Consulte *DataLogicalOffset* para obtener detalles.

El valor inicial de este campo es 0.

MQRR-Registro de respuesta

Utilice la estructura MQRR para recibir el código de terminación y el código de razón resultantes de la operación de apertura o colocación para una sola cola de destino, cuando el destino es una lista de distribución. MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1 .

Disponibilidad

La estructura MQRR está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para clientes de IBM MQ conectados a estos sistemas.

Juego de caracteres y codificación

Los datos de MQRR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en las llamadas MQOPEN y MQPUT, o en la llamada MQPUT1 , puede determinar los códigos de terminación y los códigos de razón para todas las colas de una lista de distribución cuando el resultado de la llamada se mezcla, es decir, cuando la llamada es satisfactoria para algunas colas de la lista, pero falla para otras. El código de razón MQRC_MULTIPLE_REASON de la llamada indica que el gestor de colas ha establecido los registros de respuesta (si los proporciona la aplicación).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 524. Campos en MQRR

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
CompCode (código de terminación para cola)	MQCC_OK	0
Razón (código de razón para cola)	MQRC_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQRR_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRR MyRR = {MQRR_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQRR

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
  MQLONG  CompCode;  /* Completion code for queue */
  MQLONG  Reason;    /* Reason code for queue */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para MQRR

```
dcl
```

```
 1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */
```

Declaración de Visual Basic para MQRR

```
Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type
```

CompCode (MQLONG)

Es el código de terminación resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es MQCC_OK.

Razón (MQLONG)

Este es el código de razón resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es MQRC_NONE.

MQSCO-Opciones de configuración de SSL/TLS

La estructura MQSCO, junto con los campos TLS de la estructura MQCD, permite a una aplicación que se ejecuta como IBM MQ MQI client especificar opciones de configuración que controlan el uso de TLS para la conexión de cliente cuando el protocolo de canal es TCP/IP. La estructura es un parámetro de entrada en la llamada MQCONNX.

Disponibilidad

La estructura MQSCO está disponible en los clientes siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

Si el protocolo de canal para el canal de cliente no es TCP/IP, se ignora la estructura MQSCO.

Juego de caracteres y codificación

Los datos de MQSCO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 525. Entrada de campos MQSCO		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
StrucId (identificador de estructura)	MQSCO_STRUC_ID	'SCO¬'
Versión (número de versión de estructura)	MQSCO_CURRENT_VERSION	1
KeyRepository (ubicación del repositorio de claves)	Ninguna	Serie nula o espacios en blanco
CryptoHardware (detalles del hardware criptográfico)	Ninguna	Serie nula o espacios en blanco
AuthInfoRecCount (número de registros MQAIR presentes)	Ninguna	0
AuthInfoRecOffset (desplazamiento del primer registro MQAIR desde el inicio de MQSCO)	Ninguna	0
AuthInfoRecPtr (dirección del primer registro MQAIR)	Ninguna	Puntero nulo o bytes nulos
Nota: Los dos campos siguientes se ignoran si *Version* es menor que MQSCO_VERSION_2.		
KeyResetCount (recuento de restablecimiento de clave secreta TLS)	MQSCO_RESET_COUNT_DEFAULT	0
“FipsRequired (MQLONG)” en la página 571 (utilice algoritmos criptográficos certificados por FIPS en IBM MQ)	MQSSL_FIPS_NO	0
Nota: Los dos campos siguientes se ignoran si *Version* es menor que MQSCO_VERSION_3.		
EncryptionPolicySuiteB (utilizar solo algoritmos criptográficos Suite B)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
Nota: Los dos campos siguientes se ignoran si *Version* es menor que MQSCO_VERSION_4.		
PolíticaCertificateVal (política de validación de certificados)	MQ_CERT_VAL_POLICY_DEFAULT	0
Nota: Los dos campos siguientes se ignoran si *Version* es menor que MQSCO_VERSION_5.		
CertificateLabel (detalla la etiqueta de certificado que se está utilizando)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

2. En el lenguaje de programación C, la variable de macroMQSCO_DEFAULT contiene los valores listados en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */

  };
```

Declaración COBOL para 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 **
```

Declaración PL/I para 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 */
```

Declaración de Visual Basic para 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
```

Referencia relacionada

“MQCNO - Opciones de conexión” en la página 316

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión con el gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONNX.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQSCO_ID_ESTRUCTURA
: Identificador para la estructura de opciones de configuración TLS.

  Para el lenguaje de programación C, también se define la constante MQSCO_STRUC_ID_ARRAY; tiene el mismo valor que MQSCO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSCO_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSCO_VERSION_1
Estructura de opciones de configuración de TLS Version-1 .

MQSCO_VERSION_2
Estructura de opciones de configuración de TLS Version-2 .

MQSCO_VERSION_3
Estructura de opciones de configuración de TLS Version-3 .

MQSCO_VERSION_4
Estructura de opciones de configuración de TLS Version-4 .

MQSCO_VERSION_5
Estructura de opciones de configuración de TLS Version-5 .

La constante siguiente especifica el número de versión de la versión actual:

MQSCO_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de configuración de TLS.

Siempre es un campo de entrada. El valor inicial de este campo es MQSCO_VERSION_1.

ULW *KeyRepository (MQCHAR256)*

Este campo sólo es relevante para IBM MQ MQI clients que se ejecuta en sistemas UNIX, Linux, and Windows . Especifica la ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. El archivo de base de datos de claves debe tener un nombre de archivo con el formato `zzz.kdb`, donde `zzz` es seleccionable por el usuario. El campo *KeyRepository* contiene la vía de acceso a este archivo, junto con la raíz del nombre de archivo (todos los caracteres del nombre de archivo hasta, pero sin incluir, el `.kdb`final). El sufijo de archivo `.kdb` se añade automáticamente.

Cada archivo de base de datos de claves tiene un *archivo de ocultación de contraseña*asociado. Esto contiene las contraseñas codificadas que se utilizan para permitir el acceso programático a la base de datos de claves. El archivo de ocultación de contraseña debe residir en el mismo directorio y tener la misma raíz de archivo que la base de datos de claves, y debe finalizar con el sufijo `.sth`.

Por ejemplo, si el campo *KeyRepository* tiene el valor `/xxx/yyy/key`, el archivo de base de datos de claves debe ser `/xxx/yyy/key.kdb`y el archivo de ocultación de contraseña debe ser `/xxx/yyy/key.sth`, donde `xxx` y `yyy` representan nombres de directorio.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. El valor no se comprueba; si hay un error al acceder al repositorio de claves, la llamada falla con el código de razón MQRC_KEY_REPOSITORY_ERROR.

Para ejecutar una conexión TLS desde un IBM MQ MQI client, establezca *KeyRepository* en un nombre de archivo de base de datos de claves válido.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SSL_KEY_REPOSITORY_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

CryptoHardware (MQCHAR256)

Este campo proporciona detalles de configuración para el hardware criptográfico conectado al sistema cliente.

Establezca el campo en una serie con el formato siguiente, o déjelo en blanco o nulo:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting;
```

Para utilizar hardware criptográfico que se ajuste a la interfaz PKCS #11 , por ejemplo, se debe especificar IBM 4960 o IBM 4764, la vía de acceso del controlador PKCS #11 , la etiqueta de señal PKCS #11 y las series de contraseña de señal PKCS #11 , cada una terminada con un punto y coma.

La vía de acceso del controlador PKCS #11 es una vía de acceso absoluta a la biblioteca compartida que proporciona soporte para la tarjeta PKCS #11 . El nombre del archivo de controlador PKCS #11 es el nombre de la biblioteca compartida. Un ejemplo del valor necesario para la vía de acceso PKCS #11 y el nombre de archivo es:

```
/usr/lib/pkcs11/PKCS11_API.so
```

La etiqueta de señal PKCS #11 debe coincidir con la etiqueta con la que ha configurado el hardware.

Si no es necesaria ninguna configuración de hardware criptográfico, establezca el campo en blanco o nulo.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, o se produce una anomalía cuando se utiliza para configurar el hardware de cifrado, la llamada falla con el código de razón MQRC_CRYPTO_HARDWARE_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SSL_CRYPTO_HARDWARE_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

AuthInfoRecCount (MQLONG)

Es el número de registros de información de autenticación (MQAIR) a los que se dirigen los campos *AuthInfoRecPtr* o *AuthInfoRecOffset* . Para obtener más información, consulte“MQAIR-Registro de información de autenticación” en la página 270. El valor debe ser mayor o igual que cero. Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_REC_COUNT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de información de autenticación desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo. El campo se ignora si *AuthInfoRecCount* es cero.

Puede utilizar *AuthInfoRecOffset* o *AuthInfoRecPtr* para especificar los registros MQAIR, pero no ambos; consulte la descripción del campo *AuthInfoRecPtr* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecPtr (PMQAIR)

Es la dirección del primer registro de información de autenticación. El campo se ignora si *AuthInfoRecCount* es cero.

Puede proporcionar la matriz de registros MQAIR de una de estas dos maneras:

- Utilizando el campo de puntero *AuthInfoRecPtr*

  En este caso, la aplicación puede declarar una matriz de registros MQAIR separada de la estructura MQSCO y establecer *AuthInfoRecPtr* en la dirección de la matriz.

  Considere la posibilidad de utilizar *AuthInfoRecPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).
- Utilizando el campo de desplazamiento *AuthInfoRecOffset*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga una MQSCO seguida de la matriz de registros MQAIR y establecer *AuthInfoRecOffset* en el desplazamiento del primer registro de la matriz desde el inicio de la estructura MQSCO. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

Considere la posibilidad de utilizar *AuthInfoRecOffset* para lenguajes de programación que no dan soporte al tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Sea cual sea la técnica que elija, sólo se puede utilizar uno de *AuthInfoRecPtr* y *AuthInfoRecOffset* ; la llamada falla con el código de razón MQRC_AUTH_INFO_REC_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

KeyResetCount (MQLONG)

Esto representa el número total de bytes no cifrados enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta.

El número de bytes incluye la información de control que envía el MCA.

Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. Esto es para evitar el coste de proceso de restablecimientos de clave excesivos que se producirían para valores pequeños de restablecimiento de clave secreta TLS.

Este es un campo de entrada. El valor es un número comprendido entre 0 y 999 999 999, con un valor por omisión de 0. Utilice un valor de 0 para indicar que las claves secretas nunca se renegocian.

FipsRequired (MQLONG)

IBM MQ se puede configurar con hardware criptográfico para que los módulos de criptografía utilizados sean los proporcionados por el producto de hardware; estos pueden estar certificados por FIPS a un nivel determinado en función del producto de hardware criptográfico en uso. Utilice este campo para especificar que sólo se utilicen algoritmos certificados por FIPS si la criptografía se proporciona en el software proporcionado por IBM MQ.

Cuando se instala IBM MQ , también se instala una implementación de criptografía TLS que proporciona algunos módulos certificados por FIPS.

Los valores pueden ser:

MQSSL_FIPS_NO
Éste es el valor predeterminado. Cuando se establece en este valor:

- Se puede utilizar cualquier CipherSpec soportada en una plataforma determinada.
- Si se ejecuta sin utilizar hardware de cifrado, las CipherSpecs se ejecutan utilizando la criptografía certificada FIPS 140-2 en las plataformas IBM MQ .

  Para obtener una lista de CipherSpecscertificadas por FIPS, consulte la tabla que se describe en Habilitación de CipherSpecs.

MQSSL_FIPS_YES
Cuando se establece en este valor, a menos que esté utilizando hardware criptográfico para realizar la criptografía, puede estar seguro de que

- Solo se pueden utilizar algoritmos criptográficos con certificado FIPS en la CipherSpec que se aplica a esta conexión de cliente.
- Las conexiones de canal TLS de entrada y salida sólo son satisfactorias, si se utilizan determinadas especificaciones de cifrado.

  Consulte Habilitación de CipherSpecs para obtener más información.

Nota: Siempre que sea posible, si se configuran CipherSpecs sólo para FIPS, el cliente MQI rechaza las conexiones que especifican una CipherSpec no FIPS con MQRC_SSL_INITIALIZATION_ERROR. IBM MQ

no garantiza rechazar todas las conexiones de este tipo y es responsabilidad del usuario determinar si la configuración es compatible con IBM MQ.

EncryptionPolicySuiteB(MQLONG)

Este campo especifica si se utiliza la criptografía compatible con Suite B y qué nivel de fuerza se emplea. El valor puede ser uno o varios de los siguientes:

- MQ_SUITE_B_NONE

  No se utiliza la criptografía compatible con Suite B.
- MQ_SUITE_B_128_BIT

  Se utiliza la seguridad de potencia de 128 bits de Suite B.
- MQ_SUITE_B_192_BIT

  Se utiliza la seguridad de potencia de 192 bits de la suite B.

Nota: El uso de MQ_SUITE_B_NONE con cualquier otro valor en este campo no es válido.

Política CertificateVal(MQLONG)

Este campo especifica qué tipo de política de validación de certificados se utiliza. El campo se puede establecer en uno de los valores siguientes:

MQ_CERT_VAL_POLICY_ANY
: Aplique cada una de las políticas de validación de certificados soportadas por la biblioteca de sockets seguros. Acepte la cadena de certificados si alguna de las políticas considera válida la cadena de certificados.

MQ_CERT_VAL_POLICY_RFC5280
: Aplique sólo la política de validación de certificados compatible con RFC5280 . Este valor proporciona una validación más estricta que el valor ANY, pero rechaza algunos certificados digitales más antiguos.

El valor inicial de este campo es MQ_CERT_VAL_POLICY_ANY

CertificateLabel (MQCHAR64)

Este campo proporciona detalles de la etiqueta de certificado que se está utilizando.

IBM MQ inicializa el valor por omisión para el campo *CertificateLabel* como espacios en blanco.

Esto se interpreta en tiempo de ejecución como el valor predeterminado y es compatible con versiones anteriores.

Por ejemplo, si se especifica una versión de MQSCO menor que 5.0, o se utiliza el valor predeterminado de blancos para el campo *CertificateLabel* , se utiliza el valor predeterminado preexistente de `ibmwebspheremq`*`user_id`*.

MQSD - Descriptor de suscripción

La estructura MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando. La estructura es un parámetro de entrada/salida en la llamada MQSUB.Para obtener más información, consulte Notas de uso de MQSUB.

Disponibilidad

La estructura MQSD está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux

- Solaris Solaris
- Windows Windows
- z/OS z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQSD es MQSD_VERSION_1.

Juego de caracteres y codificación

Los datos de MQSD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Suscripciones gestionadas

Si una aplicación no tiene ninguna necesidad específica de utilizar una cola determinada como destino para las publicaciones que coinciden con su suscripción, puede utilizar la característica de suscripción gestionada. Si una aplicación opta por utilizar una suscripción gestionada, el gestor de colas informa al suscriptor sobre el destino donde se envían los mensajes publicados, proporcionando un descriptor de objeto como salida de la llamada MQSUB. Para obtener más información, consulte Hobj (MQHOBJ)-entrada/salida.

Cuando se elimina la suscripción, el gestor de colas también se compromete a limpiar los mensajes que no se han recuperado del destino gestionado, en las situaciones siguientes:

- Cuando se elimina la suscripción-mediante el uso de MQCLOSE con MQCO_REMOVE_SUB-y se cierra el Hobj gestionado.
- Implícita significa que cuando se pierde la conexión con una aplicación utilizando una suscripción no duradera (MQSO_NON_DURABLE)
- Por caducidad cuando se elimina una suscripción porque ha caducado y el Hobj gestionado está cerrado.

Debe utilizar suscripciones gestionadas con suscripciones no duraderas, para que se pueda realizar esta limpieza, y para que los mensajes de las suscripciones no duraderas cerradas no ocupen espacio en el gestor de colas. Las suscripciones duraderas también pueden utilizar destinos gestionados.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQSD_STRUC_ID	'SD¬¬'
Versión (número de versión de estructura)	MQSD_VERSION_1	1
Options (opciones)	MQSO_NON_DURABLE	0
ObjectName (nombre de objeto)	Ninguna	Serie nula o espacios en blanco
AlternateUserId (ID de usuario alternativo)	Ninguna	Serie nula o espacios en blanco

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
AlternateSecurityId (ID de seguridad alternativo)	MQSID_NONE	Nulos
SubExpiry (caducidad de suscripción)	MQEI_UNLIMITED	-1
ObjectString (serie de objeto)	Ninguna	Nombres y valores definidos para MQCHARV
SubName (nombre de suscripción)	Ninguna	Nombres y valores definidos para MQCHARV
SubUserData (datos de usuario de suscripción)	Ninguna	Nombres y valores definidos para MQCHARV
SubCorrelId (ID de correlación de suscripción)	MQCI_NONE	Nulos
PubPriority (prioridad de publicación)	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken (señal de contabilidad de publicación)	MQACT_NONE	Nulos
PubAppIdentityData (datos de identidad de aplicación de publicación)	Ninguna	Serie nula o espacios en blanco
SelectionString (serie que proporciona criterios de selección)	Ninguna	Nombres y valores definidos para MQCHARV
SubLevel (nivel de suscripción)	Ninguna	1
ResObjectString (nombre de objeto largo)	Ninguna	Nombres y valores definidos para MQCHARV

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQSD_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSD MySD = {MQSD_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este es el identificador de estructura; el valor debe ser:

MQSD_STRUC_ID
Identificador de la estructura del descriptor de suscripción.

Para el lenguaje de programación C, también se define la constante MQSD_STRUC_ID_ARRAY; tiene el mismo valor que MQSD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSD_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSD_VERSION_1
Version-1 Estructura del descriptor de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

MQSD_CURRENT_VERSION
Versión actual de la estructura del descriptor de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSD_VERSION_1.

Opciones (MQLONG)

Esto proporciona opciones para controlar la acción de la llamada MQSUB.

Debe especificar al menos una de las opciones siguientes:

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

Las combinaciones que no son válidas se anotan en este tema; cualquier otra combinación es válida.

Opciones de acceso o creación: las opciones de acceso y creación controlan si se crea una suscripción o si se devuelve o modifica una suscripción existente. Debe especificar al menos una de estas opciones.

Tabla 526. Combinaciones válidas de opciones de acceso y creación

Combinación de opciones	Notas
MQSO_CREATE	Crea una suscripción si no existe. Esta combinación falla si existe la suscripción.
MQSO_RESUME	Reanuda una suscripción existente. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_RESUME	Crea una suscripción si no existe una y reanuda una que coincida, si existe. Esta combinación es útil cuando se utiliza en una aplicación que se ejecuta varias veces.
MQSO_ALTER (consulte la nota)	Reanuda una suscripción existente, alterando los campos para que coincidan con los especificados en MQSD. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_ALTER (consulte la nota)	Crea una suscripción si no existe una y reanuda una coincidente, si existe, alterando los campos para que coincidan con los especificados en el MQSD. Esta combinación es útil cuando se utiliza en una aplicación que desea asegurarse de que su suscripción está en un estado determinado antes de continuar.

Nota:

Las opciones que especifican MQSO_ALTER también pueden especificar MQSO_RESUME, pero esta combinación no tiene ningún efecto adicional para especificar solo MQSO_ALTER. MQSO_ALTER implica MQSO_RESUME, porque llamar a MQSUB para modificar una suscripción implica que la suscripción también se reanudará. Sin embargo, lo contrario no es cierto: reanudar una suscripción no implica que deba ser alterada.

MQSO_CREATE

Cree una nueva suscripción para el tema especificado. Si existe una suscripción que utiliza el mismo *SubName* , la llamada falla con MQRC_SUB_ALREADY_EXISTS. Esta anomalía se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME. *SubName* no siempre es necesario. Para obtener más detalles, consulte la descripción de ese campo.

La combinación de MQSO_CREATE con MQSO_RESUME devuelve un manejador a una suscripción preexistente para el *SubName* especificado si se encuentra una; si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

MQSO_CREATE también se puede combinar con MQSO_ALTER con un efecto similar.

MQSO_RESUME

Devuelve un descriptor de contexto a una suscripción preexistente que coincide con la especificada por *SubName*. No se realizan cambios en los atributos de suscripciones coincidentes y se devuelven en la salida de la estructura MQSD. Sólo se utilizan los siguientes campos MQSD: StrucId, Version, Options, AlternateUserId y AlternateSecurityId y SubName.

La llamada falla con el código de razón MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre de suscripción completo. Esta anomalía se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción, o si posteriormente ha sido alterado por un ID de usuario diferente, es el ID de usuario de la modificación correcta más reciente. Si se utiliza un ID AlternateUsery se permite el uso de ID de usuario alternativo para dicho usuario, el ID de usuario alternativo se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

Si existe una suscripción coincidente que se ha creado sin la opción MQSO_ANY_USERID, y el ID de usuario de la suscripción es distinto del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y se está utilizando actualmente, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

MQSO_RESUME está implícito en MQSO_ALTER por lo que no es necesario combinarlo con esa opción. Sin embargo, la combinación de las dos opciones no provoca un error.

MQSO_ALTER

Devuelve un descriptor de contexto a una suscripción preexistente con el nombre de suscripción completo que coincide con el especificado por el nombre en *SubName*. Los atributos de la suscripción que son diferentes de los especificados en el MQSD se modifican en la suscripción a menos que no se permita la modificación para dicho atributo. Los detalles se anotan en la descripción de cada atributo y se resumen en la tabla siguiente. Si intenta modificar un atributo que no se puede cambiar, o modificar una suscripción que ha establecido la opción MQSO_IMMUTABLE, la llamada falla con el código de razón que se muestra en la tabla siguiente.

La llamada falla con el código de razón MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre de suscripción completo. Puede evitar esta anomalía combinando la opción MQSO_CREATE con MQSO_ALTER.

La combinación de MQSO_CREATE con MQSO_ALTER devuelve un manejador a una suscripción preexistente para el *SubName* especificado si se encuentra una; si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción o, si posteriormente se modifica mediante un ID de usuario diferente, es el ID de usuario de la modificación más reciente y satisfactoria. Si se utiliza un ID AlternateUsery se permite el uso de ID de usuario alternativo para ese usuario, el ID de usuario alternativo se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

Si existe una suscripción coincidente que se ha creado sin la opción MQSO_ANY_USERID y el ID de usuario de la suscripción es diferente del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y se está utilizando actualmente, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

La tabla siguiente muestra la capacidad de MQSO_ALTER para modificar los valores de atributo en MQSD y MQSUB.

Tabla 527. Atributos en MQSD y MQSUB que se pueden modificar

Descriptor de tipo de datos o llamada de función	Nombre de campo	¿Se puede modificar este atributo utilizando MQSO_ALTER	Código de razón
MQSD	Opciones de durabilidad	No	MQRC_DURABILITY_NOT_ALTERABLE

Tabla 527. Atributos en MQSD y MQSUB que se pueden modificar (continuación)			
Descriptor de tipo de datos o llamada de función	**Nombre de campo**	**¿Se puede modificar este atributo utilizando MQSO_ALTER**	**Código de razón**
MQSD	Opciones de destino	Sí	Ninguna
MQSD	Opciones de registro	Sí (consulte la nota "1" en la página 580)	MQRC_GROUPING_NOT_ALTERABLE si intenta modificar MQSO_GROUP_SUB
MQSD	Opciones de publicación	Sí (consulte la nota "2" en la página 580)	Ninguna
MQSD	Opciones de comodín	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Otras opciones	No (consulte la nota "3" en la página 580)	Ninguna
MQSD	ObjectName	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	AlternateUserId	No (consulte la nota "4" en la página 580)	Ninguna
MQSD	AlternateSecurityId	No (consulte la nota "4" en la página 580)	Ninguna
MQSD	SubExpiry	Sí	Ninguna
MQSD	ObjectString	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	No (consulte la nota "5" en la página 580)	Ninguna
MQSD	SubUserData	Sí	Ninguna
MQSD	SubCorrelId	Sí (consulte la nota "6" en la página 580)	MQRC_GROUPING_NOT_ALTERABLE cuando está en una suscripción agrupada
MQSD	PubPriority	Sí	Ninguna
MQSD	PubAccountingToken	Sí	Ninguna
MQSD	PubApplIdentityData	Sí	Ninguna
MQSD	SubLevel	No	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	Hobj	Sí (consulte la nota "6" en la página 580)	MQRC_GROUPING_NOT_ALTERABLE cuando está en una suscripción agrupada

Notas:

1. No se puede modificar MQSO_GROUP_SUB.
2. MQSO_NEW_PUBLICATIONS_ONLY no se puede modificar porque no forma parte de la suscripción
3. Estas opciones no forman parte de la suscripción
4. Este atributo no forma parte de la suscripción
5. Este atributo es la identidad de la suscripción que se está alterando
6. Modificable excepto cuando forma parte de un subgrupo agrupado (MQSO_GROUP_SUB)

Opciones de durabilidad: Las opciones siguientes controlan la durabilidad de la suscripción. Sólo puede especificar una de estas opciones. Si está alterando una suscripción existente utilizando la opción MQSO_ALTER, no puede cambiar la durabilidad de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de durabilidad adecuada.

MQSO_DURABLE
Solicite que la suscripción a este tema permanezca hasta que se elimine explícitamente utilizando MQCLOSE con la opción MQCO_REMOVE_SUB. Si esta suscripción no se elimina explícitamente, permanecerá incluso después de que se cierre esta conexión de aplicaciones con el gestor de colas.

Si se solicita una suscripción duradera a un tema que está definido como que no permite suscripciones duraderas, la llamada falla con MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NON_DURABLE
Solicite que la suscripción a este tema se elimine cuando se cierre la conexión de las aplicaciones con el gestor de colas, si todavía no se ha eliminado explícitamente. MQSO_NON_DURABLE es lo contrario

de la opción MQSO_DURABLE y se define para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

Opciones de destino: La siguiente opción controla el destino al que se envían las publicaciones de un tema al que se ha suscrito. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el destino utilizado para las publicaciones de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_GESTIONADO

Solicite que el gestor de colas gestione el destino al que se envían las publicaciones.

El descriptor de objeto devuelto en *Hobj* representa una cola gestionada del gestor de colas y se utiliza con llamadas MQGET, MQCB, MQINQ o MQCLOSE posteriores.

No se puede proporcionar un descriptor de contexto de objeto devuelto desde una llamada MQSUB anterior en el parámetro **Hobj** cuando no se especifica MQSO_MANAGED.

MQSO_NO_MULTICAST

Solicite que el destino al que se envían las publicaciones no sea una dirección de grupo de multidifusión. Esta opción sólo es válida cuando se combina con la opción MQSO_MANAGED. Cuando se proporciona un descriptor de contexto para una cola en el parámetro **Hobj** , no se puede utilizar la multidifusión para esta suscripción y la opción no es válida.

Si el tema está definido para permitir sólo suscripciones de multidifusión, utilizando el valor `MCAST(ONLY)` , la llamada falla con el código de razón MQRC_MULTICAST_REQUIRED.

Opción de ámbito: La opción siguiente controla el ámbito de la suscripción que se está realizando. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, esta opción de ámbito de suscripción no se puede cambiar. Al volver de una llamada MQSUB utilizando MQSO-RESUME, se establece la opción de ámbito adecuada.

MQSO_SCOPE_QMGR

Esta suscripción sólo se realiza en el gestor de colas local. No se distribuye ninguna suscripción de proxy a otros gestores de colas de la red. Sólo las publicaciones que se publican en este gestor de colas se envían a este suscriptor. Esto altera temporalmente cualquier comportamiento establecido utilizando el atributo de tema SUBSCOPE.

Nota: Si no se establece, el ámbito de suscripción lo determina el atributo de tema SUBSCOPE.

Opciones de registro: Las opciones siguientes controlan los detalles del registro que se realiza en el gestor de colas para esta suscripción. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de registro se pueden cambiar. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establecen las opciones de registro adecuadas.

SUB_GRUPO_MQSO

Esta suscripción se va a agrupar con otras suscripciones del mismo SubLevel utilizando la misma cola y especificando el mismo ID de correlación para que cualquier publicación a temas que provoque que se proporcione más de un mensaje de publicación al grupo de suscripciones, debido a que se está utilizando un conjunto solapado de series de tema, sólo hace que se entregue un mensaje a la cola. Si no se utiliza esta opción, cada suscripción exclusiva (identificada mediante SubName) que coincida se proporciona con una copia de la publicación, lo que podría significar que más de una copia de la publicación se puede colocar en la cola compartida por un número de suscripciones.

Sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación. La suscripción más significativa se basa en el nombre de tema completo hasta el punto en el que se encuentra un comodín. Si se utiliza una mezcla de esquemas de comodín dentro del grupo, sólo es importante la posición del comodín. Se recomienda no combinar diferentes esquemas de comodín dentro de un grupo de suscripciones que comparten la misma cola.

Al crear una nueva suscripción agrupada, todavía debe tener un SubNameexclusivo, pero si coincide con el nombre de tema completo de una suscripción existente en el grupo, la llamada falla con MQRC_DUPLICATE_GROUP_SUB.

Si la suscripción más significativa del grupo también especifica MQSO_NOT_OWN_PUBS y se trata de una publicación de la misma aplicación, no se entrega ninguna publicación a la cola.

Al modificar una suscripción realizada con esta opción, los campos que implican la agrupación, Hobj en la llamada MQSUB (que representa la cola y el nombre del gestor de colas) y el ID SubCorrelno se pueden cambiar. El intento de modificarlos hace que la llamada falle con MQRC_GROUPING_NOT_ALTERABLE.

Esta opción se debe combinar con MQSO_SET_CORREL_ID con un ID de SubCorrelque no esté establecido en MQCI_NONE y no se pueda combinar con MQSO_MANAGED.

MQSO_ANY_USERID

Cuando se especifica MQSO_ANY_USERID, la identidad del suscriptor no está restringida a un ID de usuario único. Esto permite que cualquier usuario modifique o reanude la suscripción cuando disponga de la autoridad adecuada. Sólo puede tener la suscripción un único usuario a la vez. Un intento de reanudar el uso de una suscripción actualmente en uso por otra aplicación hace que la llamada falle con MQRC_SUBSCRIPTION_IN_USE.

Para añadir esta opción a una suscripción existente, la llamada MQSUB (utilizando MQSO_ALTER) debe proceder del mismo ID de usuario que la propia suscripción original.

Si una llamada MQSUB hace referencia a una suscripción existente con MSQO_ANY_USERID establecido y el ID de usuario difiere de la suscripción original, la llamada sólo será satisfactoria si el nuevo ID de usuario tiene autorización para suscribirse al tema. Al finalizar correctamente, las futuras publicaciones de este suscriptor se colocan en la cola de suscriptores con el nuevo ID de usuario establecido en el mensaje de publicación.

No especifique MQSO_ANY_USERID y MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

MQSO_FIXED_USERID

Cuando se especifica MQSO_FIXED_USERID, sólo el último ID de usuario que modifica la suscripción puede modificar o reanudar la suscripción. Si la suscripción no se ha modificado, es el ID de usuario que ha creado la suscripción.

Si un verbo MQSUB hace referencia a una suscripción existente con MQSO_ANY_USERID establecido y altera la suscripción utilizando MQSO_ALTER para utilizar la opción MQSO_FIXED_USERID, el ID de usuario de la suscripción se fija ahora en este nuevo ID de usuario. La llamada sólo es satisfactoria si el nuevo ID de usuario tiene autoridad para suscribirse al tema.

Si un ID de usuario distinto del registrado como propietario de una suscripción intenta reanudar o modificar una suscripción MQSO_FIXED_USERID, la llamada falla con MQRC_IDENTITY_MISMATCH. El ID de usuario propietario de una suscripción se puede ver mediante el mandato DISPLAY SBSTATUS.

No especifique MQSO_ANY_USERID y MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

Opciones de publicación: Las opciones siguientes controlan la forma en que se envían las publicaciones a este suscriptor. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de publicación se pueden cambiar.

MQSO_NOT_OWN_PUBS
Indica al intermediario que la aplicación no desea ver ninguna de sus propias publicaciones. Se considera que las publicaciones se originan en la misma aplicación si los manejadores de conexión son los mismos. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_NEW_PUBLICATIONS_ONLY
No se enviarán publicaciones retenidas actualmente, cuando se cree esta suscripción, sólo se enviarán nuevas publicaciones. Esta opción sólo se aplica cuando se especifica MQSO_CREATE. Cualquier cambio posterior en una suscripción no altera el flujo de publicaciones y, por lo tanto, las publicaciones retenidas en un tema, ya se habrán enviado al suscriptor como nuevas publicaciones.

Si se especifica esta opción sin MQSO_CREATE, la llamada falla con MQRC_OPTIONS_ERROR. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción no se establece incluso si la suscripción se ha creado utilizando esta opción.

Si no se utiliza esta opción, los mensajes retenidos anteriormente se envían a la cola de destino proporcionada. Si esta acción falla debido a un error, ya sea MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, la creación de la suscripción falla.

MQSO_PUBLICATIONS_ON_REQUEST
El establecimiento de esta opción indica que el suscriptor solicitará información específicamente cuando sea necesario. El gestor de colas no envía mensajes no solicitados al suscriptor. La publicación retenida (o posiblemente varias publicaciones si se especifica un comodín en el tema) se envía al suscriptor cada vez que se realiza una llamada MQSUBRQ utilizando el descriptor de contexto Hsub de una llamada MQSUB anterior. No se envían publicaciones como resultado de la llamada MQSUB utilizando esta opción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

Esta opción no es válida en combinación con un SubLevel mayor que 1.

Opciones de lectura anticipada: Las opciones siguientes controlan si los mensajes no persistentes se envían a una aplicación antes de que la aplicación los solicite.

MQSO_READ_AHEAD_AS_Q_DEF
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, el atributo de lectura anticipada predeterminado de la cola de modelo asociada con el tema al que se ha suscrito determina si los mensajes se envían a la aplicación antes de que la aplicación los solicite.

Éste es el valor predeterminado.

MQSO_NO_READ_AHEAD
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, los mensajes no se envían a la aplicación antes de que la aplicación los solicite.

MQSO_READ_AHEAD
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, es posible que se envíen mensajes a la aplicación antes de que la aplicación los solicite.

Nota:

Las siguientes notas se aplican a las opciones de lectura anticipada:

1. Sólo se puede especificar una de estas opciones. Si se especifican MQOO_READ_AHEAD y MQOO_NO_READ_AHEAD, se devuelve el código de razón MQRC_OPTIONS_ERROR. Estas opciones sólo son aplicables si se especifica MQSO_MANAGED.
2. No son aplicables para MQSUB cuando se pasa una cola que se ha abierto anteriormente. Es posible que la lectura anticipada no esté habilitada cuando se solicite. Las opciones MQGET utilizadas en la primera llamada MQGET pueden impedir que se habilite la lectura anticipada. Además, la lectura anticipada está inhabilitada cuando el cliente se está conectando a un gestor de colas donde la lectura anticipada no está soportada. Si la aplicación no se ejecuta como un cliente IBM MQ , estas opciones se ignoran.

Opciones de comodín: Las opciones siguientes controlan cómo se interpretan los comodines en la serie proporcionada en el campo ObjectString de MQSD. Sólo puede especificar una de estas opciones. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de comodín no se pueden cambiar. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de comodín adecuada.

MQSO_WILDCARD_CHAR
Los comodines sólo operan en caracteres dentro de la serie de tema.

El comportamiento definido por MQSO_WILDCARD_CHAR se muestra en la tabla siguiente.

Tabla 528. Cómo se interpretan los comodines	
Carácter especial	**Comportamiento**
Barra inclinada (/)	Sin significación, sólo otro carácter
Los caracteres de asterisco (*)	Comodín, cero o más caracteres
Un signo de interrogación (?)	Comodín, 1 carácter
Signo de porcentaje (%)	Carácter de escape para permitir que los caracteres (*), (?) o (%) se utilicen en una serie y no se interpreten como un carácter especial, por ejemplo, (% *), (%?) o (%%).

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Nota: Este uso de comodines proporciona exactamente el significado proporcionado en IBM MQ V6 y WebSphere MB V6 cuando se utilizan mensajes con formato MQRFH1 para publicación/suscripción. Se recomienda que no se utilice para las aplicaciones recién escritas y que solo se utilice para las aplicaciones que se ejecutaban anteriormente en esa versión y que no se han modificado para utilizar el comportamiento de comodín predeterminado tal como se describe en MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TOPIC

Los comodines sólo operan en elementos de tema dentro de la serie de tema. Este es el comportamiento predeterminado si no se elige ninguno.

El comportamiento necesario para MQSO_WILDCARD_TOPIC se muestra en la tabla siguiente:

Tabla 529. Cómo se interpretan los comodines	
Carácter especial	**Comportamiento**
(/)	Separador de nivel de tema
signo de almohadilla (#)	Comodín: nivel de varios temas
Signo más (+)	Comodín: nivel de tema único

Notas:

Los caracteres (+) y (#) no se tratan como comodines si se mezclan con otros caracteres (incluidos ellos mismos) dentro de un nivel de tema. En la serie siguiente, los caracteres (#) y (+) se tratan como caracteres ordinarios.

```
level0/level1/#+/level3/level#
```

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Otras opciones: Las opciones siguientes controlan la forma en que se emite la llamada de API en lugar de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, estas opciones no se modifican. Consulte "AlternateUserId (MQCHAR12)" en la página 586 para obtener más detalles.

MQSO_ALTERNATE_USER_AUTHORITY

El campo AlternateUserId contiene un identificador de usuario para utilizarlo para validar esta llamada MQSUB. La llamada sólo puede realizarse correctamente si este ID AlternateUserestá autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo.

MQSO_SET_CORREL_ID

La suscripción debe utilizar el identificador de correlación proporcionado en el campo *SubCorrelId* . Si no se especifica esta opción, el gestor de colas crea automáticamente un identificador de correlación en el momento de la suscripción y se devuelve a la aplicación en el campo *SubCorrelId* . Para obtener más información, consulte "SubCorrelId (MQBYTE24)" en la página 589 .

Esta opción no se puede combinar con MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT

La suscripción es para utilizar la señal de contabilidad y los datos de identidad de aplicación proporcionados en los campos *PubAccountingToken* y *PubApplIdentityData* .

Si se especifica esta opción, se realiza la misma comprobación de autorización que si se accediera a la cola de destino mediante una llamada MQOPEN con MQOO_SET_IDENTITY_CONTEXT, excepto en el caso en que se utilice también la opción MQSO_MANAGED, en cuyo caso no hay comprobación de autorización en la cola de destino.

Si no se especifica esta opción, las publicaciones enviadas a este suscriptor tienen información de contexto predeterminada asociada a ellos, de la manera siguiente:

Tabla 530. Información de contexto predeterminada para las publicaciones enviadas a este suscriptor

Campo de MQMD	Valor utilizado
UserIdentifier	El ID de usuario asociado a la suscripción en el momento en que se realizó la suscripción.
AccountingToken	Se determina a partir del entorno, si es posible; se establece en MQACT_NONE si no es así.
ApplIdentityData	Establecer en blancos

Esta opción sólo es válida con MQSO_CREATE y MQSO_ALTER. Si se utiliza con MQSO_RESUME, los campos *PubAccountingToken* y *PubApplIdentityData* se ignoran, por lo que esta opción no tiene ningún efecto.

Si se altera una suscripción sin utilizar esta opción donde anteriormente la información de contexto de identidad proporcionada por la suscripción, se genera la información de contexto predeterminada para la suscripción alterada.

Si una suscripción que permite que distintos ID de usuario la utilicen con la opción MQSO_ANY_USERID, se reanuda con un ID de usuario diferente, se genera contexto de identidad predeterminado para el nuevo ID de usuario que es propietario ahora de la suscripción y las publicaciones posteriores se entregan conteniendo el nuevo contexto de identidad.

MQSO_FAIL_IF_QUIESCING

La llamada MQSUB falla si el gestor de colas está en estado de desactivación temporal. En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQSUB falle si la conexión está en estado de desactivación temporal.

ObjectName (MQCHAR48)

Es el nombre del objeto de tema tal como se ha definido en el gestor de colas local.

El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el nulo y los caracteres que le siguen se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS:
  - Evite los nombres que empiezan o terminan con un guión bajo; no pueden ser procesados por las operaciones y los paneles de control.
  - El carácter de porcentaje tiene un significado especial para RACF. Si se utiliza RACF como gestor de seguridad externa, los nombres no deben contener el porcentaje. Si lo hacen, estos nombres no se incluyen en ninguna comprobación de seguridad cuando se utilizan perfiles genéricos de RACF .
- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

ObjectName se utiliza para formar el nombre de tema completo.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

Si no se puede encontrar el objeto identificado por el campo *ObjectName* , la llamada falla con el código de razón MQRC_UNKNOWN_OBJECT_NAME incluso si hay una serie especificada en *ObjectString*.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no cambia.

La longitud de este campo la proporciona MQ_TOPIC_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre del objeto de tema al que se ha suscrito no se puede cambiar. Este campo y el campo *ObjectString* se pueden omitir. Si se proporcionan, deben resolverse en el mismo nombre de tema completo. Si no lo hacen, la llamada falla con MQRC_TOPIC_NOT_ALTERABLE.

AlternateUserId (MQCHAR12)

Si especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la suscripción y para la salida en la cola de destino (especificada en el parámetro **Hobj** de la llamada MQSUB), en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación.

Si es satisfactorio, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se ejecuta actualmente la aplicación.

Si se especifica MQSO_ALTERNATE_USER_AUTHORITY y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción sólo puede realizarse correctamente si no se necesita ninguna autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

Existen las siguientes diferencias en los entornos indicados:

- En z/OS, sólo se utilizan los primeros 8 caracteres del ID AlternateUserpara comprobar la autorización para la suscripción. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario sólo debe contener caracteres permitidos por el gestor de seguridad externa.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo no se modifica.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

AlternateSecurityId (MQBYTE40)

Este es un identificador de seguridad que se transfiere con el AlternateUserId al servicio de autorizaciones para permitir que se realicen las comprobaciones de autorización correspondientes.

AlternateSecurityId sólo se utiliza si se especifica MQSO_ALTERNATE_USER_AUTHORITY y el campo AlternateUserId no está completamente en blanco hasta el primer carácter nulo o el final del campo.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo no se modifica.

Consulte la descripción de “AlternateSecurityId (MQBYTE40)” en la página 494 en el tipo de datos MQOD para obtener más información.

SubExpiry (MQLONG)

Este es el tiempo expresado en décimas de segundo tras el cual caduca la suscripción. No habrá más publicaciones que coincidan con esta suscripción después de que haya pasado este intervalo. Tan pronto como caduca una suscripción, las publicaciones ya no se envían a la cola. Sin embargo, las publicaciones que ya están allí no se ven afectadas de ninguna manera. *SubExpiry* no tiene ningún efecto en la caducidad de la publicación.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED
La suscripción tiene un tiempo de caducidad ilimitado.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar la caducidad de la suscripción.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo se establece en la caducidad original de la suscripción y no en el tiempo de caducidad restante.

ObjectString (MQCHARV)

Es el nombre de objeto largo que se va a utilizar.

ObjectString se utiliza para formar el nombre de tema completo.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

La longitud máxima de *ObjectString* es 10240.

Si *ObjectString* no se especifica correctamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_OBJECT_STRING_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si hay comodines en *ObjectString* , la interpretación de dichos comodines se puede controlar utilizando las opciones de comodín especificadas en el campo Opciones de MQSD.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no cambia. El nombre de tema completo utilizado se devuelve en el campo *ResObjectString* si se proporciona un almacenamiento intermedio.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre largo del objeto de tema al que se ha suscrito no se puede cambiar. Este campo y el campo *ObjectName* se pueden omitir. Si se proporcionan, deben resolverse en el mismo nombre de tema completo o la llamada falla con MQRC_TOPIC_NOT_ALTERABLE.

SubName (MQCHARV)

Especifica el nombre de suscripción. Este campo sólo es necesario si *Options* especifica la opción MQSO_DURABLE, pero si se proporciona también lo utilizará el gestor de colas para MQSO_NON_DURABLE.

Si se especifica, *SubName* debe ser exclusivo dentro del gestor de colas, porque es el método utilizado para identificar la suscripción.

La longitud máxima de *SubName* es 10240.

Este campo sirve para dos propósitos. Para una suscripción MQSO_DURABLE, utilice este campo para identificar una suscripción para poder reanudarla después de que se haya creado si ha cerrado el descriptor de contexto de la suscripción (utilizando la opción MQCO_KEEP_SUB) o se ha desconectado del gestor de colas. Esto se realiza utilizando la llamada MQSUB con la opción MQSO_RESUME. También se visualiza en la vista administrativa de suscripciones en el campo SUBID en DISPLAY SBSTATUS.

Si *SubName* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , se deja de lado cuando es necesario (es decir, *SubName*. *VSLength* es cero), o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SUB_NAME_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre de suscripción no se puede cambiar, porque es el campo de identificación utilizado para encontrar la suscripción referenciada. No se cambia en la salida de una llamada MQSUB con la opción MQSO_RESUME.

Datos de SubUser(MQCHARV)

Especifica los datos de usuario de suscripción. Los datos proporcionados en la suscripción en este campo se incluirán como la propiedad de mensaje MQSubUserData de cada publicación que se envíe a esta suscripción.

La longitud máxima de *SubUserData* es 10240.

Si *SubUserData* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SUB_USER_DATA_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, los datos de usuario de suscripción se pueden cambiar.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio y hay una longitud de almacenamiento intermedio positiva en *VSBufLen*. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devuelve la longitud de la fecha de usuario de suscripción en el campo *VSLength* de MQCHARV. Si el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven *VSBufLen* bytes en el almacenamiento intermedio proporcionado.

SubCorrelId (MQBYTE24)

Este campo contiene un identificador de correlación común a todas las publicaciones que coinciden con esta suscripción.

Atención: un identificador de correlación sólo se puede pasar entre gestores de colas en un clúster de publicación/suscripción, no en una jerarquía.

Todas las publicaciones enviadas para que coincidan con esta suscripción contienen este identificador de correlación en el descriptor de mensaje. Si varias suscripciones obtienen sus publicaciones de la misma cola, el uso de MQGET por identificador de correlación sólo permite obtener publicaciones para una suscripción específica. Este identificador de correlación puede ser generado por el gestor de colas o por el usuario.

Si no se especifica la opción MQSO_SET_CORREL_ID, el gestor de colas genera el identificador de correlación y este campo es un campo de salida que contiene el identificador de correlación que se establecerá en cada mensaje publicado para esta suscripción. El identificador de correlación generado consta de un identificador de producto de 4 bytes (AMQX o CSQM en ASCII o EBCDIC) seguido de una implementación específica de producto de una serie exclusiva.

Si se especifica la opción MQSO_SET_CORREL_ID, el usuario genera el identificador de correlación y este campo es un campo de entrada que contiene el identificador de correlación que se debe establecer en cada publicación para esta suscripción. En este caso, si el campo contiene MQCI_NONE, el identificador de correlación que se establece en cada mensaje publicado para esta suscripción es el identificador de correlación creado por la colocación original del mensaje.

Si la opción MQSO_GROUP_SUB se ha especificado y el identificador de correlación especificado es el mismo que una suscripción agrupada existente que utiliza la misma cola y una serie de tema que se solapa, sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación.

La longitud de este campo la proporciona MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Si está alterando una suscripción existente utilizando la opción MQSO_ALTER, y este campo es un campo de entrada, el identificador de correlación de suscripción se puede cambiar, a menos que la suscripción sea una suscripción agrupada, es decir, se ha creado utilizando la opción MQSO_GROUP_SUB, en cuyo caso el identificador de correlación de suscripción no se puede cambiar.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el identificador de correlación actual para la suscripción.

PubPriority (MQLONG)

Este es el valor que estará en el campo *Priority* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. Para obtener más información sobre el campo *Priority* en MQMD, consulte “Priority (MQLONG)” en la página 452.

El valor debe ser mayor o igual que cero, donde cero es la prioridad más baja. También se pueden utilizar los valores especiales siguientes:

MQPRI_PRIORITY_AS_Q_DEF

Cuando se proporciona una cola de suscripción en el campo *Hobj* de la llamada MQSUB, y no es un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo **DefPriority** de esta cola. Si la cola es una cola de clúster o hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad se determina cuando el mensaje de publicación se coloca en la cola tal como se describe para "Priority (MQLONG)" en la página 452.

Si la llamada MQSUB utiliza un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo **DefPriority** de la cola modelo asociada al tema al que se ha suscrito.

MQPRI_PRIORITY_AS_PUBLISHED

La prioridad del mensaje es la prioridad de la publicación original. Es el valor inicial del campo.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el *Priority* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en la prioridad actual que se utiliza para la suscripción.

Señal PubAccounting(MQBYTE32)

Este es el valor que estará en el campo *AccountingToken* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. *AccountingToken* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje. Para obtener más información sobre el campo *AccountingToken* en MQMD, consulte "AccountingToken (MQBYTE32)" en la página 461

Puede utilizar el siguiente valor especial para el campo *PubAccountingToken* :

MQACT_NONE

No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQACT_NONE_ARRAY también está definida; tiene el mismo valor que MQACT_NONE, pero es una matriz de caracteres en lugar de una serie.

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el gestor de colas genera la señal de contabilidad como información de contexto predeterminada y este campo es un campo de salida que contiene el *AccountingToken* que se establecerá en cada mensaje publicado para esta suscripción.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el usuario está generando la señal de contabilidad y este campo es un campo de entrada que contiene el *AccountingToken* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial de este campo es MQACT_NONE.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el valor de *AccountingToken* en cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el *AccountingToken* actual que se utiliza para la suscripción.

PubApplIdentityData (MQCHAR32)

Es el valor que se encuentra en el campo *ApplIdentityData* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *ApplIdentityData* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje,

consulte Contexto de mensaje. Para obtener más información sobre el campo *ApplIdentityData* en MQMD, consulte “Datos de ApplIdentity(MQCHAR32)” en la página 462

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el *ApplIdentityData* que se establece en cada mensaje publicado para esta suscripción está en blanco, como información de contexto predeterminada.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el usuario está generando *PubApplIdentityData* y este campo es un campo de entrada que contiene el *ApplIdentityData* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 32 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el *ApplIdentityData* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el *ApplIdentityData* actual que se utiliza para la suscripción.

SelectionString (MQCHARV)

Es la serie utilizada para proporcionar los criterios de selección utilizados al suscribirse a mensajes de un tema.

Este campo de longitud variable se devolverá en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio, y también hay una longitud de almacenamiento intermedio positiva en VSBufSize. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devolverá la longitud de la serie de selección en el campo VSLength de MQCHARV. Si el búfer proporcionado es inferior al espacio necesario para devolver el campo, solo se devuelven VSBufSize bytes en el búfer.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura “MQCHARV-Serie de longitud variable” en la página 293 , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SELECTION_STRING_ERROR.

El uso de SelectionString se describe en Selectores.

SubLevel (MQLONG)

Éste es el nivel asociado a la suscripción. Las publicaciones sólo se entregan a esta suscripción si está en el conjunto de suscripciones con el valor de SubLevel más alto menor o igual que el valor de PubLevel utilizado en el momento de la publicación. Sin embargo, si una publicación se ha retenido, ya no está disponible para los suscriptores en niveles superiores porque se vuelve a publicar en PubLevel 1.

El valor debe estar en el rango de cero a 9. Cero sería el nivel más bajo.

El valor inicial de este campo es 1.

Para obtener más información, consulte Interceptación de publicaciones.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el SubLevel no se puede cambiar.

No se permite combinar un SubLevel con un valor mayor que 1 con la opción MQSO_PUBLICATIONS_ON_REQUEST.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el nivel actual que se utiliza para la suscripción.

Serie ResObject(MQCHARV)

Es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en *ObjectName*.

Si el nombre de objeto largo se proporciona en *ObjectString* y no se proporciona nada en *ObjectName*, el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si este campo se omite (es decir, ResObjectString.VSBufSize es cero), no se devuelve *ResObjectString* , pero la longitud se devuelve en ResObjectString.VSLength. Si la longitud es menor que la serie ResObjectcompleta, se trunca y devuelve tantos caracteres situados más a la derecha como caben en la longitud proporcionada.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_RES_OBJECT_STRING_ERROR.

MQSMPO-Establecer opciones de propiedad de mensaje

La estructura **MQSMPO** permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada **MQSETMP** .

Disponibilidad

Todos los sistemas IBM MQ y clientes IBM MQ .

Juego de caracteres y codificación

Los datos de **MQSMPO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 531. Campos en MQSMPO

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQSMPO_STRUC_ID	'SMPO'
Versión (número de versión de estructura)	MQSMPO_VERSION_1	1
Options (opciones)	MQSMPO_NONE	0
ValueEncoding (codificación de valor de propiedad)	MQENC_NATIVE	Depende del entorno
ValueCCSID (juego de caracteres de valor de propiedad)	MQCCSI_APPL	-3

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQSMPO_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este es el identificador de estructura; el valor debe ser:

MQSMPO_STRUC_ID

Identificador para establecer la estructura de opciones de propiedad de mensaje.

Para el lenguaje de programación C, la constante **MQSMPO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQSMPO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQSMPO_STRUC_ID**.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSMPO_VERSION_1
Version-1 establece la estructura de opciones de propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQSMPO_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de establecer propiedad de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es **MQSMPO_VERSION_1**.

Opciones (MQLONG)

Opciones de ubicación

Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de propiedad:

MQSMPO_SET_FIRST
Establece el valor de la primera propiedad que coincide con el nombre especificado, o si no existe, añade una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente.

MQSMPO_SET_PROP_UNDER_CURSOR
Establece el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR
Establece una nueva propiedad antes de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_AFTER_CURSOR
Establece una nueva propiedad después de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_APPEND_PROPERTY
Hace que se añada una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente. Si existe al menos una propiedad que coincide con el nombre especificado, se añade una nueva propiedad al final después del final de la lista de propiedades.

Esta opción permite crear una lista de propiedades con el mismo nombre.

Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQSMPO_NONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSMPO_SET_FIRST.

ValueEncoding (MQLONG)

La codificación del valor de propiedad que se va a establecer si el valor es numérico.

Siempre es un campo de entrada. El valor inicial de este campo es **MQENC_NATIVE**.

ValueCCSID (MQLONG)

El juego de caracteres del valor de propiedad que se va a establecer si el valor es una serie de caracteres.

Siempre es un campo de entrada. El valor inicial de este campo es **MQCCSI_APPL**.

MQSRO-Opciones de solicitud de suscripción

La estructura MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción. La estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Disponibilidad

La estructura MQSRO está disponible en las plataformas siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQSRO es MQSRO_VERSION_1.

Juego de caracteres y codificación

Los datos de MQSRO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQSRO_STRUC_ID	'SRO¬'
Versión (número de versión de estructura)	MQSRO_VERSION_1	1
Options (opciones)	MQSRO_NONE	0
NumPubs (número de publicaciones)	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQSRO_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Este es el identificador de estructura; el valor debe ser:

MQSRO_STRUC_ID
Identificador de la estructura de opciones de solicitud de suscripción.

Para el lenguaje de programación C, también se define la constante MQSRO_STRUC_ID_ARRAY; tiene el mismo valor que MQSRO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSRO_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSRO_VERSION_1
Version-1 Estructura de opciones de solicitud de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

MQSRO_CURRENT_VERSION
Versión actual de la estructura de opciones de solicitud de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSRO_VERSION_1.

Opciones (MQLONG)

Debe especificarse una de las opciones siguientes. Sólo se puede especificar una opción.

MQSRO_FAIL_IF_QUIESCING

La llamada MQSUBRQ falla si el gestor de colas está en estado de desactivación temporal. En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQSUBRQ falle si la conexión está en un estado de desactivación temporal.

Opción predeterminada: Si la opción descrita anteriormente no es necesaria, se debe utilizar la opción siguiente:

MQSRO_NONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQSRO_NONE ayuda a la documentación del programa. Aunque no se pretende que esta opción se utilice con ninguna otra, debido a que su valor es cero, no se puede detectar este uso.

NumPubs (MQLONG)

Es un campo de salida, devuelto a la aplicación para indicar el número de publicaciones enviadas a la cola de suscripción como resultado de esta llamada. Aunque este número de publicaciones se han enviado como resultado de esta llamada, no hay garantía de que este número de mensajes estén disponibles para que la aplicación los obtenga, especialmente si son mensajes no persistentes.

Puede haber más de una publicación si el tema al que se ha suscrito contenía un comodín. Si no había comodines en la serie de tema cuando se creó la suscripción representada por *Hsub* , como máximo se envía una publicación como resultado de esta llamada.

MQSTS-Estructura de informes de estado

La estructura MQSTS es un parámetro de salida del mandato MQSTAT. El mandato MQSTAT se utiliza para recuperar información de estado. Esta información se devuelve en una estructura MQSTS.

Juego de caracteres y codificación

Los datos de tipo carácter en MQSTS están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQSTS están en la codificación de máquina nativa; esto se proporciona mediante *Codificación*.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 532. Campos en MQSTS

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQSTS_ID_STRUCT	'STAT¬'
Versión (número de versión de estructura)	MQSTS_VERSION_1	1
CompCode (código de terminación del primer error)	MQCC_OK	0
Razón (código de razón del primer error)	MQRC_NONE	0
PutSuccessCount (número de llamadas put asíncronas satisfactorias)	Ninguna	0
PutWarningCount (número de llamadas de colocación asíncronas que tenían avisos)	Ninguna	0
PutFailureRecuento (número de llamadas de colocación asíncronas fallidas)	Ninguna	0
ObjectType (tipo de objeto anómalo)	MQOT_Q	1
ObjectName (nombre del objeto anómalo)	Ninguna	Serie nula o espacios en blanco
ObjectQMgrNombre (nombre del gestor de colas propietario del objeto anómalo	Ninguna	Serie nula o espacios en blanco
ResolvedObjectNombre (nombre resuelto de la cola de destino)	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrNombre (nombre resuelto del gestor de colas de destino)	Ninguna	Serie nula o espacios en blanco
Nota: Los campos restantes se ignoran si la versión es menor que MQSTS_VERSION_2.		
ObjectString (nombre de objeto largo de objeto anómalo)	MQCHARV_PREDETERMINADO	{NULL,0,0,0,-3}
SubName (nombre de suscripción de suscripción anómala)	MQCHARV_PREDETERMINADO	{NULL,0,0,0,-3}
OpenOptions (opciones de apertura asociadas a la anomalía)	Ninguna	0
SubOptions(opciones de suscripción asociadas con el error)	Ninguna	0

Tabla 532. Campos en MQSTS (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macro MQSTS_DEFAULT contiene los valores que se listan en la tabla. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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 **
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Referencia relacionada

"MQSTAT-Recuperar información de estado" en la página 796
Utilice la llamada MQSTAT para recuperar información de estado. El tipo de información de estado devuelta viene determinado por el valor de tipo especificado en la llamada.

StrucId (MQCHAR4)

Identificador de la estructura de informes de estado, MQSTS.

StrucId es el identificador de estructura. El valor debe ser:

MQSTS_ID_STRUCT
Identificador de la estructura de informes de estado.

Para el lenguaje de programación C, la constante MQSTS_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQSTS_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucId es siempre un campo de entrada. Su valor inicial es MQSTS_STRUC_ID.

Versión (MQLONG)

El número de versión de la estructura.

El valor debe ser:

MQSTS_VERSION_1
Estructura de informes de estado de la versión 1.

MQSTS_VERSION_2
Estructura de informes de estado de la versión 2.

La constante siguiente especifica el número de versión de la versión actual:

MQSTS_CURRENT_VERSION
Versión actual de la estructura de informes de estado. La versión actual es MQSTS_VERSION_2.

Version es siempre un campo de entrada. Su valor inicial es MQSTS_VERSION_1.

CompCode (MQLONG)

El código de terminación de la operación sobre la que se informa.

La interpretación de CompCode depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Es el código de terminación resultante de una operación de colocación asíncrona anterior en el objeto especificado en `ObjectName`.

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando o no se ha podido volver a conectar, este es el código de finalización que ha hecho que la conexión empezara a volver a conectarse.

Si la conexión está conectada actualmente, el valor es `MQCC_OK`.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no se ha podido volver a conectar, este es el código de terminación que ha hecho que la reconexión fallara.

Si la conexión está conectada actualmente, o se vuelve a conectar, el valor es `MQCC_OK`.

`CompCode` es siempre un campo de salida. Su valor inicial es `MQCC_OK`.

Razón (MQLONG)

El código de razón de la operación sobre la que se informa.

La interpretación de `Reason` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Este es el código de razón resultante de una operación de colocación asíncrona anterior en el objeto especificado en `ObjectName`.

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando o no se ha podido reconectar, este es el código de razón que ha hecho que la reconexión empezara a reconectarse.

Si la conexión está conectada actualmente, el valor es `MQRC_NONE`.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el código de razón que ha hecho que la reconexión fallara.

Si la conexión está conectada actualmente, o se vuelve a conectar, el valor es `MQRC_NONE`.

`Reason` es un campo de salida. Su valor inicial es `MQRC_NONE`.

Recuento de PutSuccess(MQLONG)

El número de operaciones de colocación asíncronas que se han realizado correctamente.

El valor de `PutSuccessCount` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura `MQSTS` que se ha completado con `MQCC_OK`.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

`PutSuccessCount` es un campo de salida. Su valor inicial es cero.

Recuento de PutWarning(MQLONG)

Número de operaciones de colocación asíncronas que han finalizado con un aviso.

El valor de PutWarningCount depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura MQSTS que se ha completado con MQCC_WARNING.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

PutWarningCount es un campo de salida. Su valor inicial es cero.

Recuento de PutFailure(MQLONG)

El número de operaciones de colocación asíncrona que han fallado.

El valor de PutFailureCount depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura MQSTS que se ha completado con MQCC_FAILED.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

PutFailureCount es un campo de salida. Su valor inicial es cero.

ObjectType (MQLONG)

El tipo del objeto especificado en *ObjectName* sobre el que se informa.

Los valores posibles de ObjectType se listan en "MQOT_ * (Tipos de objeto y tipos de objeto ampliados)" en la página 164.

ObjectType es un campo de salida. Su valor inicial es MQOT_Q.

ObjectName (MQCHAR48)

El nombre del objeto sobre el que se informa.

La interpretación de ObjectName depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre de la cola o tema utilizado en la operación de colocación, cuya anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando, este es el nombre del gestor de colas asociado a la conexión.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el nombre del objeto que ha hecho que fallara la reconexión. La razón de la anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

ObjectName es un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)

El nombre del gestor de colas sobre el que se informa.

La interpretación de `ObjectQMgrName` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre del gestor de colas en el que se define el objeto *`ObjectName`* . Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

V 9.1.3 **MQSTAT_TYPE_RECONNECTION**

Multi

El campo **ObjectQMgrName** contiene el nombre de un gestor de colas con el que se solicita la reconexión, o en blanco si no se especifica ningún gestor de colas. Si es posible, el cliente intenta volver a conectarse a un gestor de colas con ese nombre.

z/OS

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el nombre del objeto que ha hecho que fallara la reconexión. La razón de la anomalía se notifica en los campos *`CompCode`* y *`Reason`* de la estructura `MQSTS` .

`ObjectQMgrName` es un campo de salida. Su valor es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedObjectNombre (MQCHAR48)

El nombre del objeto nombrado en *`ObjectName`* después de que el gestor de colas local resuelva el nombre.

La interpretación de `ResolvedObjectName` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedObjectName` es el nombre del objeto nombrado en *`ObjectName`* después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de un objeto que existe en el gestor de colas identificado por *`ResolvedQMgrName`*.

MQSTAT_TYPE_RECONNECTION

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

En blanco.

`ResolvedObjectName` es un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQMgrNombre (MQCHAR48)

El nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre.

La interpretación de `ResolvedQMgrName` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedQMgrName` es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas que es propietario del objeto identificado por *`ResolvedObjectName`*. *`ResolvedQMgrName`* puede ser el nombre del gestor de colas local.

MQSTAT_TYPE_RECONNECTION

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

En blanco.

ResolvedQMgrName es siempre un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectString (MQCHARV)

Nombre de objeto largo del objeto anómalo sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de ObjectString depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre de objeto largo de la cola o tema utilizado en la operación MQPUT , que ha fallado.

MQSTAT_TYPE_RECONNECTION

Serie de caracteres de longitud cero

MQSTAT_TYPE_RECONNECTION_ERROR

Es el nombre de objeto largo del objeto que ha hecho que fallara la reconexión.

ObjectString es un campo de salida. Su valor inicial es una serie de longitud cero.

SubName (MQCHARV)

El nombre de la suscripción anómala. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de SubName depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Serie de longitud cero.

MQSTAT_TYPE_RECONNECTION

Serie de longitud cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El nombre de la suscripción que ha hecho que fallara la reconexión. Si no hay ningún nombre de suscripción disponible, o la anomalía no está relacionada con una suscripción, se trata de una serie de longitud cero.

SubName es un campo de salida. Su valor inicial es una serie de longitud cero.

OpenOptions (MQLONG)

El OpenOptions utilizado para abrir el objeto sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

El valor de OpenOptions depende del valor del parámetro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El OpenOptions utilizado cuando se produjo la anomalía. La razón de la anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

OpenOptions es un campo de salida. Su valor inicial es cero.

SubOptions (MQLONG)

El `SubOptions` utilizado para abrir la suscripción anómala. Sólo está presente en la versión 2 de `MQSTS` o superior.

La interpretación de `SubOptions` depende del valor del parámetro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El `SubOptions` utilizado cuando se produjo la anomalía. Si la anomalía no está relacionada con la suscripción a un tema, el valor devuelto es cero.

`SubOptions` es un campo de salida. Su valor inicial es cero.

MQTM-Mensaje de desencadenante

La estructura MQTM describe los datos del mensaje desencadenante que envía el gestor de colas a una aplicación de supervisor desencadenante cuando se produce un suceso desencadenante para una cola.Esta estructura forma parte de IBM MQ Trigger Monitor Interface (TMI), que es una de las interfaces de infraestructura de IBM MQ .

Nombre de formato

MQFMT_TRIGGER.

Juego de caracteres y codificación

Los datos de caracteres en MQTM están en el juego de caracteres del gestor de colas que genera MQTM. Los datos numéricos en MQTM están en la codificación de máquina del gestor de colas que genera MQTM.

El juego de caracteres y la codificación de MQTM se proporcionan mediante los campos *`CodedCharSetId`* y *`Encoding`* en:

- El MQMD (si la estructura MQTM está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQTM (todos los demás casos).

Utilización

Es posible que una aplicación de supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación que inicia la aplicación de supervisor desencadenante. La información que puede necesitar la aplicación iniciada incluye *`QName`*, *`TriggerData`*y *`UserData`*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada. Para obtener información sobre MQTMC2, consulte “MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)” en la página 613.

- z/OS En z/OS, para una aplicación MQAT_CICS que se inicia utilizando la transacción CKTI, toda la estructura de mensajes desencadenantes MQTM está disponible para la transacción iniciada; la información se puede recuperar utilizando el mandato EXEC CICS RETRIEVE.
- IBM i En IBM i, la aplicación de supervisor desencadenante proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Para obtener información sobre cómo utilizar desencadenantes, consulte Inicio de aplicaciones IBM MQ utilizando desencadenantes.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 533. Campos en MQTM para MQTM

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQTM_STRUC_ID	'TM¬¬'
Versión (número de versión de estructura)	MQTM_VERSION_1	1
QName (nombre de cola desencadenada)	Ninguna	Serie nula o espacios en blanco
ProcessName (nombre del objeto de proceso)	Ninguna	Serie nula o espacios en blanco
TriggerData (datos de desencadenante)	Ninguna	Serie nula o espacios en blanco
ApplType (tipo de aplicación	Ninguna	0
ApplId (identificador de aplicación)	Ninguna	Serie nula o espacios en blanco
EnvData (datos de entorno)	Ninguna	Serie nula o espacios en blanco
UserData (datos de usuario)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQTM_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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 para un mensaje desencadenante

Tabla 534. Valores para los campos en el MQMD de un mensaje desencadenante generado por el gestor de colas

Campo de MQMD	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	MQMT_DATAGRAM
Expiry	MQEI_UNLIMITED
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	Atributo **CodedCharSetId** del gestor de colas
Format	MQFMT_TRIGGER
Priority	Atributo **DefPriority** de la cola de inicio
Persistence	MQPER_NOT_PERSISTENT
MsgId	Un valor exclusivo
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Nombre de gestor de colas
UserIdentifier	Espacios en blanco
AccountingToken	MQACT_NONE
ApplIdentityData	Espacios en blanco
PutApplType	MQAT_QMGR, o según corresponda para el agente de canal de mensajes
PutApplName	Primeros 28 bytes del nombre del gestor de colas
PutDate	Fecha en la que se envía el mensaje desencadenante
PutTime	Hora a la que se envía el mensaje desencadenante
ApplOriginData	Espacios en blanco

Se recomienda una aplicación que genere un mensaje desencadenante para establecer valores similares, excepto para lo siguiente:

- El campo *Priority* se puede establecer en MQPRI_PRIORITY_AS_Q_DEF (el gestor de colas lo cambiará por la prioridad predeterminada para la cola de inicio cuando se coloque el mensaje).
- El campo *ReplyToQMgr* se puede establecer en blancos (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando coloque el mensaje).
- Establezca los campos de contexto según corresponda para la aplicación.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQTM_STRUC_ID
Identificador de la estructura del mensaje desencadenante.

Para el lenguaje de programación C, también se define la constante MQTM_STRUC_ID_ARRAY; tiene el mismo valor que MQTM_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQTM_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQTM_VERSION_1
Número de versión para la estructura del mensaje desencadenante.

La constante siguiente especifica el número de versión de la versión actual:

MQTM_CURRENT_VERSION
Versión actual de la estructura del mensaje desencadenante.

El valor inicial de este campo es MQTM_VERSION_1.

QName (MQCHAR48)

Es el nombre de la cola para la que se ha producido un suceso desencadenante y lo utiliza la aplicación iniciada por la aplicación de supervisor desencadenante. El gestor de colas inicializa este campo con el valor del atributo **QName** de la cola desencadenada; consulte"Atributos para colas" en la página 848para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ProcessName (MQCHAR48)

Es el nombre del objeto de proceso del gestor de colas especificado para la cola desencadenada y puede ser utilizado por la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ProcessName** de la cola identificada por el campo *QName* ; consulte"Atributos para colas" en la página 848para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo siempre se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona MQ_PROCESS_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

TriggerData (MQCHAR64)

Se trata de datos de formato libre para que los utilice la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **TriggerData** de la cola identificada por el campo *QName* ; consulte "Atributos para colas" en la página 848 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

En z/OS, para una aplicación CICS iniciada utilizando la transacción CKTI, esta información no se utiliza.

La longitud de este campo la proporciona MQ_TRIGGER_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 64 caracteres en blanco en otros lenguajes de programación.

ApplType (MQLONG)

Esto identifica la naturaleza del programa que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ApplType** del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 885 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

ApplType puede tener uno de los siguientes valores estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a los valores del rango MQAT_USER_FIRST a MQAT_USER_LAST:

MQAT_AIX
Aplicación AIX (mismo valor que MQAT_UNIX).

MQAT_LOTE
aplicación por lotes

INTERMEDIARIO
Aplicación de intermediario

MQAT_CICS
Transacción CICS .

MQAT_CICS_PUENTE
CICS bridge .

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
Aplicación IBM MQ MQI client en PC DOS.

MQAT_IMS
IMS .

MQAT_IMS_BRIDGE
Aplicación puente IMS .

MQAT_JAVA
Java .

MQAT_MVS
Aplicación MVS o TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Aplicación de agente.

MQAT_OS390
Aplicación OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
IBM i .

MQAT_RRS_BATCH
Aplicación por lotes RRS.

MQAT_UNIX
UNIX .

MQAT_DESCONOCIDO
Aplicación de tipo desconocido.

USUARIO_MQ
Tipo de aplicación definido por el usuario.

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
z/OS .

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

El valor inicial de este campo es `0`.

ApplId (MQCHAR256)

Es una serie de caracteres que identifica la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ApplId** del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 885 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *ApplId* sea el nombre de un programa ejecutable. Las notas siguientes se aplican a los entornos indicados:

- En z/OS, *ApplId* es:
  - Un identificador de transacción CICS , para las aplicaciones iniciadas utilizando la transacción CKTI del supervisor desencadenante de CICS
  - Un identificador de transacción IMS , para las aplicaciones iniciadas utilizando el supervisor desencadenante de IMS CSQQTRMN
- En sistemas Windows , el nombre de programa puede tener como prefijo una unidad y una vía de acceso de directorio.
- En IBM i, el nombre de programa puede tener como prefijo un nombre de biblioteca y/carácter.
- En UNIX, el nombre de programa puede tener como prefijo una vía de acceso de directorio.

La longitud de este campo la proporciona MQ_PROCESS_APPL_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 256 caracteres en blanco en otros lenguajes de programación.

EnvData (MQCHAR128)

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **EnvData** del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 885 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

En z/OS, para una aplicación CICS iniciada utilizando la transacción CKTI, o una aplicación IMS que se va a iniciar utilizando la transacción CSQQTRMN, esta información no se utiliza.

La longitud de este campo la proporciona MQ_PROCESS_ENV_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 128 caracteres en blanco en otros lenguajes de programación.

UserData (MQCHAR128)

Es una serie de caracteres que contiene información de usuario relevante para la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **UserData** del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 885 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

Para Microsoft Windows, la serie de caracteres no debe contener comillas dobles si la definición de proceso se va a pasar a **runmqtrm**.

La longitud de este campo la proporciona MQ_PROCESS_USER_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 128 caracteres en blanco en otros lenguajes de programación.

MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)

Cuando una aplicación de supervisor desencadenante recupera un mensaje desencadenante (MQTM) de una cola de inicio, es posible que el supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación que inicia el supervisor desencadenante.

La información que puede necesitar la aplicación iniciada incluye *QName*, *TriggerData*y *UserData*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada.

Esta estructura forma parte de IBM MQ Trigger Monitor Interface (TMI), que es una de las interfaces de infraestructura de IBM MQ .

Juego de caracteres y codificación

Los datos de tipo carácter en MQTMC2 están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId** .

Utilización

La estructura MQTMC2 es muy similar al formato de la estructura MQTM. La diferencia es que los campos que no son de tipo carácter en MQTM se cambian en MQTMC2 por campos de tipo carácter de la misma longitud, y el nombre del gestor de colas se añade al final de la estructura.

- z/OS En z/OS, para una aplicación MQAT_IMS que se inicia utilizando la aplicación CSQQTRMN, se pone a disposición de la aplicación iniciada una estructura MQTMC2 .
- IBM i En IBM i, la aplicación de supervisor desencadenante proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 535. Campos en MQTMC2

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQTMC_STRUC_ID	'TMC¬'
Versión (número de versión de estructura)	MQTMC_VERSION_2	'¬¬¬2'
QName (nombre de cola desencadenada)	Ninguna	Serie nula o espacios en blanco
ProcessName (nombre del objeto de proceso)	Ninguna	Serie nula o espacios en blanco
TriggerData (datos de desencadenante)	Ninguna	Serie nula o espacios en blanco
ApplType (tipo de aplicación)	Ninguna	Espacios en blanco
ApplId (identificador de aplicación)	Ninguna	Serie nula o espacios en blanco

Tabla 535. Campos en MQTMC2 (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
EnvData (datos de entorno)	Ninguna	Serie nula o espacios en blanco
UserData (datos de usuario)	Ninguna	Serie nula o espacios en blanco
QMgrName (nombre de gestor de colas)	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQTMC2_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Identificador de estructura.

El valor debe ser:

MQTMC_STRUC_ID
Identificador de la estructura del mensaje desencadenante (formato de caracteres).

Para el lenguaje de programación C, también se define la constante MQTMC_STRUC_ID_ARRAY; tiene el mismo valor que MQTMC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQCHAR4)

Número de versión de la estructura.

El valor debe ser:

MQTMC_VERSION_2
Estructura del mensaje desencadenante de la versión 2 (formato de caracteres).

Para el lenguaje de programación C, también se define la constante MQTMC_VERSION_2_ARRAY ; tiene el mismo valor que MQTMC_VERSION_2, pero es una matriz de caracteres en lugar de una serie.

La constante siguiente especifica el número de versión de la versión actual:

MQTMC_CURRENT_VERSION
Versión actual de la estructura del mensaje desencadenante (formato de caracteres).

QName (MQCHAR48)
Nombre de la cola desencadenada.

Consulte el campo *QName* en la estructura MQTM.

ProcessName (MQCHAR48)
Nombre del objeto de proceso.

Consulte el campo *ProcessName* en la estructura MQTM.

TriggerData (MQCHAR64)
Datos del desencadenante.

Consulte el campo *TriggerData* en la estructura MQTM.

ApplType (MQCHAR4)
Tipo de aplicación.

Este campo siempre contiene espacios en blanco, cualquiera que sea el valor del campo *ApplType* en la estructura MQTM del mensaje desencadenante original.

ApplId (MQCHAR256)

Identificador de aplicación.

Consulte el campo *ApplId* en la estructura MQTM.

EnvData (MQCHAR128)
Datos de entorno.

Consulte el campo *EnvData* en la estructura MQTM.

UserData (MQCHAR128)
Datos de usuario.

Consulte el campo *UserData* en la estructura MQTM.

QMgrName (MQCHAR48)
Nombre de gestor de colas.

Es el nombre del gestor de colas en el que se ha producido el suceso desencadenante.

MQWIH - Cabecera de información de trabajo

Si el gestor de carga de trabajo (WLM) de z/OS va a procesar un mensaje, el mensaje debe empezar con una estructura MQWIH. Esta estructura describe la información que debe estar presente al principio de un mensaje que debe ser manejado por WLM.

Disponibilidad

Todos los sistemas IBM MQ , más los clientes IBM MQ conectados a estos sistemas.

Nombre de formato

MQFMT_WORK_INFO_HEADER.

Juego de caracteres y codificación

Los campos de la estructura MQWIH están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* en la estructura de cabecera que precede a MQWIH, o por esos campos en la estructura MQMD si MQWIH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Utilización

Para cualquier plataforma soportada por IBM MQ , puede crear y transmitir un mensaje que incluya la estructura MQWIH, pero sólo un gestor de colas IBM MQ for z/OS puede interactuar con WLM. Por lo tanto, para que el mensaje llegue a WLM desde un gestor de colas que no seaz/OS , la red del gestor de colas debe incluir al menos un gestor de colas z/OS a través del cual se pueda direccionar el mensaje.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 536. Campos en MQWIH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQWIH_STRUC_ID	'WIH¬'
Versión (número de versión de estructura)	MQWIH_VERSION_1	1
StrucLength (longitud de la estructura MQWIH)	MQWIH_LENGTH_1	120
Codificación (codificación numérica de datos que sigue a MQWIH)	Ninguna	0
CodedCharSetId (identificador de juego de caracteres de los datos que siguen a MQWIH)	MQCCSI_UNDEFINED	0
Formato (nombre de formato de los datos que siguen a MQWIH)	MQFMT_NONE	Espacios en blanco
Distintivos (distintivos)	MQWIH_NONE	0
ServiceName (nombre de servicio)	Ninguna	Espacios en blanco
ServiceStep (nombre de paso de servicio)	Ninguna	Espacios en blanco
MsgToken (señal de mensaje)	MQMTOK_NONE	Nulos
Reservado (reservado)	Ninguna	Espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQWIH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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)

Es el identificador de estructura. El valor debe ser:

MQWIH_STRUC_ID
Identificador de la estructura de cabecera de información de trabajo.

Para el lenguaje de programación C, también se define la constante MQWIH_STRUC_ID_ARRAY; tiene el mismo valor que MQWIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQWIH_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQWIH_VERSION_1
Estructura de cabecera de información de trabajo Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQWIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es MQWIH_VERSION_1.

StrucLength (MQLONG)

Es la longitud de la estructura MQWIH. El valor debe ser:

MQWIH_LENGTH_1
Longitud de la estructura de cabecera de información de trabajo version-1 .

La constante siguiente especifica la longitud de la versión actual:

MQWIH_CURRENT_LENGTH
Longitud de la versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es MQWIH_LENGTH_1.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a la estructura MQWIH; no se aplica a los datos numéricos de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQWIH; no se aplica a los datos de tipo carácter de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
: Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

  El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

  MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a la estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor debe ser:

MQWIH_NONE
: Sin distintivos.

El valor inicial de este campo es MQWIH_NONE.

ServiceName (MQCHAR32)

Es el nombre del servicio que va a procesar el mensaje.

La longitud de este campo la proporciona MQ_SERVICE_NAME_LENGTH. El valor inicial de este campo es de 32 caracteres en blanco.

ServiceStep (MQCHAR8)

Es el nombre del paso de *ServiceName* con el que se relaciona el mensaje.

La longitud de este campo la proporciona MQ_SERVICE_STEP_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

MsgToken (MQBYTE16)

Se trata de una señal de mensaje que identifica de forma exclusiva el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo se ignora. La longitud de este campo la proporciona MQ_MSG_TOKEN_LENGTH. El valor inicial de este campo es MQMTOK_NONE.

Reservado (MQCHAR32)

Es un campo reservado; debe estar en blanco.

MQXP-Bloque de parámetros de salida

La estructura MQXP se utiliza como parámetro de entrada/salida para la salida cruzada de API.Para obtener más información sobre esta salida, consulte La salida cruzada de API.

Juego de caracteres y codificación

Los datos de tipo carácter en MQXP se encuentran en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId** . Los datos numéricos en MQXP están en la codificación de máquina nativa; esto lo proporciona MQENC_NATIVE.

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 537. Campos en MQXP

Nombre de campo y descripción	Nombre de constante
StrucId (identificador de estructura)	MQXP_STRUC_ID
Versión (número de versión de estructura)	MQXP_VERSION_1
ExitId (identificador de salida)	MQXT_API_CROSSING_EXIT
ExitReason (razón de la invocación de la salida)	Ninguna
ExitResponse (respuesta de salida)	Ninguna
ExitCommand (código de llamada de API)	Ninguna
ExitParmCount (recuento de parámetros)	Ninguna
Reservado (reservado)	Ninguna
ÁreaExitUser (área de usuario)	Ninguna

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)

Es el identificador de estructura. El valor debe ser:

MQXP_STRUC_ID
Identificador de la estructura de parámetros de salida.

Para el lenguaje de programación C, también se define la constante MQXP_STRUC_ID_ARRAY; tiene el mismo valor que MQXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQXP_VERSION_1
Número de versión para la estructura de bloque de parámetros de salida.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

ExitId (MQLONG)

Se establece en la entrada de la rutina de salida e indica el tipo de salida:

MQXT_API_CROSSING_EXIT
Salida cruzada de API para CICS.

Es un campo de entrada para la salida.

ExitReason (MQLONG)

Se establece en la entrada de la rutina de salida. Para la salida cruzada de API, indica si se llama a la rutina antes o después de la ejecución de la llamada de API:

MQXR_ANTES
Antes de la ejecución de la API.

MQXR_AFTER
Después de la ejecución de la API.

Es un campo de entrada para la salida.

ExitResponse (MQLONG)

El valor lo establece la salida para comunicarse con el llamante. Los valores siguientes están definidos:

MQXCC_Correcto
La salida se ha completado correctamente.

MQXCC_SUPPRESS_FUNCTION
Suprimir función.

Cuando este valor se establece mediante una salida cruzada de API llamada *antes* de la llamada de API, la llamada de API no se realiza. El *CompCode* para la llamada se establece en MQCC_FAILED, el *Reason* se establece en MQRC_SUPPRESSED_BY_EXIT y todos los demás parámetros permanecen como la salida que los deja.

Cuando este valor se establece mediante una salida cruzada de API denominada *después* de la llamada de API, el gestor de colas lo ignora.

MQXCC_SKIP_FUNCTION
Omitir función.

Cuando este valor se establece mediante una salida cruzada de API llamada *antes* de la llamada de API, la llamada de API no se realiza; *CompCode* y *Reason* y todos los demás parámetros permanecen como la salida que los deja.

Cuando este valor se establece mediante una salida cruzada de API denominada *después* de la llamada de API, el gestor de colas lo ignora.

Es un campo de salida de la salida.

ExitCommand (MQLONG)

Este campo se establece en la entrada de la rutina de salida. Identifica la llamada de API que ha hecho que se invocara la salida:

MQXC_CALLBACK
La llamada CALLBACK.

MQXC_MQBACK
Llamada MQBACK.

MQXC_MQCB
La llamada MQCB.

MQXC_MQCLOSE
Llamada MQCLOSE.

MQXC_MQCMIT
Llamada MQCMIT.

MQXC_MQCTL
La llamada MQCTL.

MQXC_MQGET
La llamada MQGET.

MQXC_MQINQ
Llamada MQINQ.

MQXC_MQOPEN
La llamada MQOPEN.

MQXC_MQPUT
La llamada MQPUT.

MQXC_MQPUT1
La llamada MQPUT1 .

MQXC_MQSET
La llamada MQSET.

MQXC_MQSTAT
Llamada MQSTAT.

MQXC_MQSUB
Llamada MQSUB.

MQXC_MQSUBRQ
Llamada MQSUBRQ.

Es un campo de entrada para la salida.

Recuento de ExitParm(MQLONG)

Este campo se establece en la entrada de la rutina de salida. Contiene el número de parámetros que toma la llamada MQ .

Tabla 538. Número de parámetros para cada llamada MQ

Nombre de llamada	Número de parámetros
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Es un campo de entrada para la salida.

Reservado (MQLONG)

Este es un campo reservado. Su valor no es significativo para la salida.

Área ExitUser(MQBYTE16)

Este es un campo que está disponible para que lo utilice la salida. Se inicializa en cero binario para la longitud del campo antes de la primera invocación de la salida para la tarea, y posteriormente los cambios realizados en este campo por la salida se conservan entre las invocaciones de la salida. Se define el valor siguiente:

MQXUA_NONE
No hay información de usuario.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY; tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH. Es un campo de entrada/salida para la salida.

MQXQH-Cabecera de cola de transmisión

La estructura MQXQH describe la información que se añade como prefijo a los datos de mensaje de aplicación de los mensajes cuando están en colas de transmisión.Una cola de transmisión es un tipo especial de cola local que contiene temporalmente mensajes destinados a colas remotas (es decir, destinados a colas que no pertenecen al gestor de colas local). Una cola de transmisión se indica mediante el atributo de cola **Usage** que tiene el valor MQUS_TRANSMISSION.

Nombre de formato

MQFMT_XMIT_Q_HEADER

Juego de caracteres y codificación

Los datos de MQXQH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de MQXQH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD separado (si la estructura MQXQH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQXQH (todos los demás casos).

Campos

Nota: En la tabla siguiente, los campos se agrupan por uso en lugar de por orden alfabético. Los temas hijo siguen la misma secuencia.

Tabla 539. Campos en MQXQH para MQXQH

Nombre de campo y descripción	Nombre de constante	Valor inicial (si existe) de constante
StrucId (identificador de estructura)	MQXQH_STRUC_ID	'XQH¬'
Versión (número de versión de estructura)	MQXQH_VERSION_1	1
RemoteQName (nombre de la cola de destino)	Ninguna	Serie nula o espacios en blanco

Tabla 539. Campos en MQXQH para MQXQH (continuación)		
Nombre de campo y descripción	**Nombre de constante**	**Valor inicial (si existe) de constante**
RemoteQMgrNombre (nombre del gestor de colas de destino)	Ninguna	Serie nula o espacios en blanco
MsgDesc (descriptor de mensaje original)	Los mismos nombres y valores que MQMD; consulte Tabla 500 en la página 423	-

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQXQH_DEFAULT contiene los valores que se listan en la tabla. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para 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 */
};
```

Declaración COBOL para 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).
```

Declaración PL/I para 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 */
```

Declaración de High Level Assembler para 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)
```

Declaración de Visual Basic para 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
```

Campos en el descriptor de mensaje separado

Un mensaje que está en una cola de transmisión tiene *dos* descriptores de mensaje:

- Un descriptor de mensaje se almacena por separado de los datos de mensaje; esto se denomina *descriptor de mensaje independiente*y lo genera el gestor de colas cuando el mensaje se coloca en la cola de transmisión. Algunos de los campos del descriptor de mensaje separado se copian del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

  El descriptor de mensaje separado es el que se devuelve a la aplicación en el parámetro **MsgDesc** de la llamada MQGET cuando se elimina el mensaje de la cola de transmisión.
- Un segundo descriptor de mensaje se almacena dentro de la estructura MQXQH como parte de los datos del mensaje; esto se denomina *descriptor de mensaje incorporado*y es una copia del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 (con variaciones menores).

El descriptor de mensaje incorporado es siempre un MQMD version-1 . Si el mensaje colocado por la aplicación tiene valores no predeterminados para uno o más de los campos version-2 en MQMD, una estructura MQMDE sigue a la MQXQH y, a su vez, va seguida de los datos del mensaje de la aplicación (si los hay). El MQMDE es:

– Generado por el gestor de colas (si la aplicación utiliza un MQMD version-2 para transferir el mensaje), o
– Ya está presente al principio de los datos del mensaje de aplicación (si la aplicación utiliza un MQMD version-1 para transferir el mensaje).

El descriptor de mensaje incorporado es el que se devuelve a la aplicación en el parámetro **MsgDesc** de la llamada MQGET cuando se elimina el mensaje de la cola de destino final.

Los campos del descriptor de mensaje separado los establece el gestor de colas tal como se muestra. Si el gestor de colas no da soporte al MQMD version-2 , se utiliza un MQMD version-1 sin pérdida de función.

Tabla 540. Valores utilizados para campos en el MQMD independiente

Campo en MQMD separado	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	Se copia desde el descriptor de mensaje incorporado, pero con los bits identificados por MQRO_ACCEPT_UNSUP_IF_XMIT_MASK establecidos en cero. (Esto impide que se genere un mensaje de informe COA o COD cuando se coloca un mensaje en una cola de transmisión o se elimina de ella.)
MsgType	Copiado del descriptor de mensaje incorporado.
Expiry	Copiado del descriptor de mensaje incorporado.
Feedback	Copiado del descriptor de mensaje incorporado.
Encoding	MQENC_NATIVE (ver nota)
CodedCharSetId	Atributo **CodedCharSetId** del gestor de colas.
Format	MQFMT_XMIT_Q_HEADER
Priority	Copiado del descriptor de mensaje incorporado.
Persistence	Copiado del descriptor de mensaje incorporado.
MsgId	El gestor de colas genera un nuevo valor. Este identificador de mensaje es diferente del *MsgId* que el gestor de colas puede haber generado para el descriptor de mensaje incorporado descrito anteriormente.
CorrelId	*MsgId* del descriptor de mensaje incorporado. Para los mensajes que se transfieren a SYSTEM.CLUSTER.TRANSMIT.QUEUE, *CorrelId* está reservado para uso interno.
BackoutCount	0
ReplyToQ	Copiado del descriptor de mensaje incorporado.
ReplyToQMgr	Copiado del descriptor de mensaje incorporado.
UserIdentifier	Copiado del descriptor de mensaje incorporado.
AccountingToken	Copiado del descriptor de mensaje incorporado. Para los mensajes que se transfieren a SYSTEM.CLUSTER.TRANSMIT.QUEUE, *AccountingToken* está reservado para uso interno.
ApplIdentityData	Copiado del descriptor de mensaje incorporado.
PutApplType	MQAT_QMGR

Tabla 540. Valores utilizados para campos en el MQMD independiente (continuación)

Campo en MQMD separado	Valor utilizado
PutApplName	Primeros 28 bytes del nombre del gestor de colas.
PutDate	Fecha en la que se colocó el mensaje en la cola de transmisión.
PutTime	Hora en que se colocó el mensaje en la cola de transmisión.
ApplOriginData	Espacios en blanco
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- En Windows, el valor de MQENC_NATIVE para Micro Focus COBOL difiere del valor de C. El valor del campo *Encoding* en el descriptor de mensaje separado es siempre el valor para C en estos entornos; este valor es 546 en decimal. Además, los campos enteros de la estructura MQXQH están en la codificación que corresponde a este valor (la codificación Intel nativa).

Campos en el descriptor de mensaje incorporado

Los campos del descriptor de mensaje incorporado tienen los mismos valores que los del parámetro **MsgDesc** de la llamada MQPUT o MQPUT1 , excepto los siguientes:

- El campo *Version* siempre tiene el valor MQMD_VERSION_1.
- Si el campo *Priority* tiene el valor MQPRI_PRIORITY_AS_Q_DEF, se sustituye por el valor del atributo **DefPriority** de la cola.
- Si el campo *Persistence* tiene el valor MQPER_PERSISTENCE_AS_Q_DEF, se sustituye por el valor del atributo **DefPersistence** de la cola.
- Si el campo *MsgId* tiene el valor MQMI_NONE, o se ha especificado la opción MQPMO_NEW_MSG_ID, o el mensaje es un mensaje de lista de distribución, *MsgId* se sustituye por un nuevo identificador de mensaje generado por el gestor de colas.

  Cuando un mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños colocados en diferentes colas de transmisión, el campo *MsgId* de cada uno de los nuevos descriptores de mensajes incorporados es el mismo que el del mensaje de lista de distribución original.
- Si se ha especificado la opción MQPMO_NEW_CORREL_ID, *CorrelId* se sustituye por un nuevo identificador de correlación generado por el gestor de colas.
- Los campos de contexto se establecen según lo indicado por las opciones MQPMO_ * _CONTEXT especificadas en el parámetro **PutMsgOpts** ; los campos de contexto son:
  - *AccountingToken*
  - *ApplIdentityData*
  - *ApplOriginData*
  - *PutApplName*
  - *PutApplType*
  - *PutDate*
  - *PutTime*
  - *UserIdentifier*
- Los campos version-2 (si estaban presentes) se eliminan del MQMD y se mueven a una estructura MQMDE, si uno o varios de los campos version-2 tienen un valor no predeterminado.

Transferir mensajes a colas remotas

Cuando una aplicación coloca un mensaje en una cola remota (especificando directamente el nombre de la cola remota o utilizando una definición local de la cola remota), el gestor de colas local:

- Crea una estructura MQXQH que contiene el descriptor de mensaje incorporado
- Añade un MQMDE si se necesita uno y todavía no está presente
- Añade los datos del mensaje de aplicación
- Coloca el mensaje en una cola de transmisión adecuada

Colocación de mensajes directamente en colas de transmisión

Una aplicación también puede colocar un mensaje directamente en una cola de transmisión. En este caso, la aplicación debe prefijar los datos del mensaje de aplicación con una estructura MQXQH e inicializar los campos con los valores adecuados. Además, el campo *Format* del parámetro **MsgDesc** de la llamada MQPUT o MQPUT1 debe tener el valor MQFMT_XMIT_Q_HEADER.

Los datos de tipo carácter de la estructura MQXQH creada por la aplicación deben estar en el juego de caracteres del gestor de colas local (definido por el atributo de gestor de colas **CodedCharSetId**), y los datos enteros deben estar en la codificación de máquina nativa. Además, los datos de tipo carácter de la estructura MQXQH deben rellenarse con espacios en blanco hasta la longitud definida del campo; los datos no deben finalizarse prematuramente utilizando un carácter nulo, porque el gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQXQH.

Sin embargo, el gestor de colas no comprueba que esté presente una estructura MQXQH o que se hayan especificado valores válidos para los campos.

Las aplicaciones no deben colocar sus mensajes directamente en SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Obtención de mensajes de colas de transmisión

Las aplicaciones que obtienen mensajes de una cola de transmisión deben procesar la información en la estructura MQXQH de una forma adecuada. La presencia de la estructura MQXQH al principio de los datos del mensaje de aplicación se indica mediante el valor MQFMT_XMIT_Q_HEADER que se devuelve en el campo *Format* del parámetro **MsgDesc** de la llamada MQGET. Los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** indican el juego de caracteres y la codificación de los datos de caracteres y enteros en la estructura MQXQH. El juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos *CodedCharSetId* y *Encoding* en el descriptor de mensaje incorporado.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQXQH_STRUC_ID
: Identificador de la estructura de cabecera de cola de transmisión.

  Para el lenguaje de programación C, también se define la constante MQXQH_STRUC_ID_ARRAY; tiene el mismo valor que MQXQH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQXQH_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQXQH_VERSION_1
: Número de versión para la estructura de cabecera de cola de transmisión.

La constante siguiente especifica el número de versión de la versión actual:

MQXQH_CURRENT_VERSION
Versión actual de la estructura de cabecera de cola de transmisión.

El valor inicial de este campo es MQXQH_VERSION_1.

RemoteQName (MQCHAR48)

Este es el nombre de la cola de mensajes que es el destino final aparente del mensaje (esto podría no ser el destino final si, por ejemplo, esta cola se define en *RemoteQMgrName* para que sea una definición local de otra cola remota).

Si el mensaje es un mensaje de lista de distribución (es decir, el campo *Format* del descriptor de mensaje incorporado es MQFMT_DIST_HEADER), *RemoteQName* está en blanco.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

RemoteQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas o del grupo de compartición de colas que es propietario de la cola que es el destino aparente final del mensaje.

Si el mensaje es un mensaje de lista de distribución, *RemoteQMgrName* está en blanco.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MsgDesc (MQMD1)

Este es el descriptor de mensaje incorporado y es una copia cercana del MQMD del descriptor de mensaje que se ha especificado como parámetro **MsgDesc** en la llamada MQPUT o MQPUT1 cuando el mensaje se colocó originalmente en la cola remota.

Nota: Se trata de un MQMD de version-1 .

Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQMD.

Llamadas de funciones

Esta sección proporciona información sobre todas las llamadas MQI que son posibles. Se proporcionan descripciones, sintaxis, información de parámetros, notas de uso e invocaciones de idioma para cada idioma posible para cada una de las diferentes llamadas.

Referencia relacionada

z/OS Ejemplos de salida CEDF de llamadas MQI

Descripciones de llamadas

En esta sección se describen las llamadas MQI.

- “MQBACK-Retrotraer cambios” en la página 635
- “MQBEGIN-Iniciar unidad de trabajo” en la página 639
- “MQBUFMH - Convertir almacenamiento intermedio en descriptor de contexto de mensaje” en la página 642
- “MQCB-Gestionar devolución de llamada” en la página 646
- “MQCB_FUNCTION-Función de devolución de llamada” en la página 656
- “MQCLOSE-Cerrar objeto” en la página 657
- “MQCMIT-Confirmar cambios” en la página 666
- “MQCONN-Conectar gestor de colas” en la página 670
- “MQCONNX - Conectar gestor de colas (ampliado)” en la página 677
- “MQCRTMH-Crear manejador de mensajes” en la página 683

- “MQCTL-devoluciones de llamada de control” en la página 686
- “MQDISC-Desconectar gestor de colas” en la página 693
- “MQDLTMH-Suprimir descriptor de mensaje” en la página 697
- “MQDLTMP-Suprimir propiedad de mensaje” en la página 699
- “Mensaje MQGET - get” en la página 702
- “MQINQ-Consultar atributos de objeto” en la página 715
- “MQINQMP-Consultar propiedad de mensaje” en la página 732
- “MQMHBUF-Convertir descriptor de mensaje en almacenamiento intermedio” en la página 738
- “MQOPEN-Abrir objeto” en la página 742
- “MQPUT-Colocar mensaje” en la página 760
- “MQPUT1 -Colocar un mensaje” en la página 774
- “MQSET - Establecer atributos de objeto” en la página 785
- “MQSETMP-Establecer propiedad de mensaje” en la página 791
- “MQSTAT-Recuperar información de estado” en la página 796
- “MQMHBUF-Convertir descriptor de mensaje en almacenamiento intermedio” en la página 738
- “MQSUB - Registrar suscripción” en la página 799
- “MQSUBRQ-Solicitud de suscripción” en la página 807

La ayuda en línea en la plataforma UNIX , en forma de páginas de *man* , está disponible para estas llamadas.

Nota: Las llamadas asociadas con la conversión de datos, MQXCNVC y MQ_DATA_CONV_EXIT, se encuentran en “salida de conversión de datos” en la página 923.

Convenciones utilizadas en las descripciones de llamada

Para cada llamada, esta colección de temas proporciona una descripción de los parámetros y el uso de la llamada en un formato que es independiente del lenguaje de programación. Esto va seguido de invocaciones típicas de la llamada, y declaraciones típicas de sus parámetros, en cada uno de los lenguajes de programación soportados.

Importante: Al codificar llamadas de API de IBM MQ , debe asegurarse de que se proporcionan todos los parámetros relevantes (tal como se describe en las secciones siguientes). No hacerlo puede producir resultados imprevisibles.

La descripción de cada llamada contiene las secciones siguientes:

Nombre de llamada
El nombre de la llamada, seguido de una breve descripción de la finalidad de la llamada.

Parámetros
Para cada parámetro, el nombre va seguido de su tipo de datos entre paréntesis () y uno de los siguientes:

entrada
Proporcione información en el parámetro cuando realice la llamada.

salida
El gestor de colas devuelve información en el parámetro cuando la llamada se completa o falla.

entrada/salida
Proporcione información en el parámetro cuando realice la llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Por ejemplo:

Compcode (MQLONG)-salida

En algunos casos, el tipo de datos es una estructura. En todos los casos, hay más información sobre el tipo de datos o la estructura en “Tipos de datos elementales” en la página 235.

Los dos últimos parámetros de cada llamada son un código de terminación y un código de razón. El código de terminación indica si la llamada se ha completado correctamente, parcialmente o no. En el código de razón se proporciona más información sobre el éxito parcial o el fracaso de la llamada. Para obtener más información sobre cada código de terminación y razón, consulte “Códigos de retorno” en la página 889.

Notas de uso
Información adicional sobre la llamada, describiendo cómo utilizarla y cualquier restricción sobre su uso.

Invocación de lenguaje ensamblador
Invocación típica de la llamada, y declaración de sus parámetros, en lenguaje ensamblador.

Invocación en C
Invocación típica de la llamada, y declaración de sus parámetros, en C.

Invocación en COBOL
Invocación típica de la llamada, y declaración de sus parámetros, en COBOL.

Invocación en PL/I
Invocación típica de la llamada, y declaración de sus parámetros, en PL/I.

Todos los parámetros se pasan por referencia.

Invocación en Visual Basic
Invocación típica de la llamada, y declaración de sus parámetros, en Visual Basic.

Otras convenciones de notación son:

Constantes
Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQOO_OUTPUT. A continuación se muestra un conjunto de constantes que tienen el mismo prefijo: MQIA_ *. Consulte “Constantes” en la página 61 para ver el valor de una constante.

Matrices
En algunas llamadas, los parámetros son matrices de series de caracteres que no tienen tamaños fijos. En las descripciones de estos parámetros, una n minúscula representa una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico que necesite.

Utilización de las llamadas en el lenguaje C

Los parámetros que son *sólo de entrada* y de tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG se pasan por valor. Para todos los demás parámetros, la *dirección* del parámetro se pasa por valor.

No es necesario que especifique todos los parámetros que se pasan por dirección cada vez que invoca una función. Cuando no necesite un parámetro determinado, especifique un puntero nulo como parámetro en la invocación de función, en lugar de la dirección de los datos de parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la llamada; en terminología C, esto significa que todas las llamadas devuelven `void`.

Declaración del parámetro Buffer

Las llamadas **MQGET**, **MQPUT**y **MQPUT1** tienen cada una un parámetro que tiene un tipo de datos no definido: el parámetro *Buffer* . Utilice este parámetro para enviar y recibir los datos de mensaje de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero normalmente es más conveniente declararlos como la estructura particular que describe el diseño de los datos en el mensaje. El prototipo de función declara el parámetro como un puntero a void, de modo que puede especificar la dirección de cualquier tipo de datos como parámetro en la invocación de la llamada.

Puntero a vacío es un puntero a datos de formato no definido. Se define como:

```
typedef void *PMQVOID;
```

MQBACK-Retrotraer cambios

La llamada MQBACK indica al gestor de colas que se deben restituir todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización.

Los mensajes colocados como parte de una unidad de trabajo se suprimen; los mensajes recuperados como parte de una unidad de trabajo se restablecen en la cola.

- En z/OS, esta llamada sólo la utilizan los programas por lotes (incluidos los programas DL/I por lotes IMS).

Sintaxis

MQBACK (*Hconn*, *CódigoComp*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_OUTCOME_PENDING
(2124, X'84C') El resultado de la operación de devolución está pendiente.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón

Notas de uso

1. Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
   - Una unidad de trabajo local, donde los cambios sólo afectan a los recursos de MQ .
   - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de MQ .

   Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte “MQBEGIN-Iniciar unidad de trabajo” en la página 639.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
   - En z/OS, utilice las llamadas siguientes:
     - Los programas por lotes (incluidos los programas DL/I por lotes IMS) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de MQ como a los recursos que pertenecen a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por z/OS Recoverable Resource Service (RRS). La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
     - Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT ROLLBACK` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS .
     - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como `ROLB` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).

- En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.

3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC-Desconectar gestor de colas" en la página 693 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   El gestor de colas mantiene *tres* conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:
   - La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).
5. La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

   Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

   La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo.

   El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

   Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción MQPMO_LOGICAL_ORDER descrita en "MQPMO-Opciones de transferencia de mensajes" en la página 503y la opción MQGMO_LOGICAL_ORDER descrita en "MQGMO-Opciones de obtención de mensajes" en la página 367.

   Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo.
6. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro **Hconn** descrito en "MQCONN-Conectar gestor de colas" en la página 670 para obtener información sobre el ámbito de los manejadores de conexión.
7. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
8. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse de esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs**

en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Invocación en C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQBEGIN-Iniciar unidad de trabajo

La llamada MQBEGIN inicia una unidad de trabajo coordinada por el gestor de colas y que puede implicar a gestores de recursos externos.

Sintaxis

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

Hconn debe ser un descriptor de conexión no compartido. Si se especifica un descriptor de conexión compartido, la llamada falla con el código de razón MQRC_HCONN_ERROR. Consulte la descripción de las opciones MQCNO_HANDLE_SHARE_ * en "MQCNO - Opciones de conexión" en la página 316 para obtener más información sobre los descriptores de contexto compartidos y no compartidos.

BeginOptions

Tipo: MQBO-entrada/salida

Estas son opciones que controlan la acción de MQBEGIN, tal como se describe en "MQBO-Opciones de inicio" en la página 277.

Si no se necesitan opciones, los programas escritos en C o S/390 assembler pueden especificar una dirección de parámetro nula, en lugar de especificar la dirección de una estructura MQBO.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_NO_EXTERNAL_PARTICIPANTES
(2121, X'849 ') No se ha registrado ningún gestor de recursos participante.

MQRC_PARTICIPANT_NOT_AVAILABLE
(2122, X'84A') El gestor de recursos participante no está disponible.

Si *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_BO_ERROR
(2134, X'856 ') Estructura de opciones de inicio no válida.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_IN_PROGRESS
(2128, X'850 ') Unidad de trabajo ya iniciada.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Notas de uso

1. Utilice la llamada MQBEGIN para iniciar una unidad de trabajo coordinada por el gestor de colas y que puede implicar cambios en los recursos propiedad de otros gestores de recursos. El gestor de colas da soporte a tres tipos de unidad de trabajo:
   - **Unidad de trabajo local coordinada por el gestor de colas**: unidad de trabajo en la que el gestor de colas es el único gestor de recursos que participa y, por lo tanto, el gestor de colas actúa como coordinador de unidad de trabajo.
     - Para iniciar este tipo de unidad de trabajo, especifique la opción MQPMO_SYNCPOINT o MQGMO_SYNCPOINT en la primera llamada MQPUT, MQPUT1o MQGET en la unidad de trabajo.
     - Para confirmar o restituir este tipo de unidad de trabajo, utilice la llamada MQCMIT o MQBACK.
   - **Unidad de trabajo global coordinada por el gestor de colas**: unidad de trabajo en la que el gestor de colas actúa como coordinador de unidad de trabajo, tanto para recursos de MQ *como para* para recursos que pertenecen a otros gestores de recursos. Estos gestores de recursos cooperan con el gestor de colas para asegurarse de que todos los cambios en los recursos de la unidad de trabajo se confirman o se restituyen juntos.
     - Para iniciar este tipo de unidad de trabajo, utilice la llamada MQBEGIN.
     - Para confirmar o restituir este tipo de unidad de trabajo, utilice las llamadas MQCMIT y MQBACK.
   - **Unidad de trabajo global coordinada externamente**: una unidad de trabajo en la que el gestor de colas es un participante, pero el gestor de colas no actúa como coordinador de unidad de trabajo. En su lugar, hay un coordinador de unidad de trabajo externo con el que el gestor de colas coopera.

– Para iniciar este tipo de unidad de trabajo, utilice la llamada pertinente proporcionada por el coordinador externo de la unidad de trabajo.

  Si se utiliza la llamada MQBEGIN para intentar iniciar la unidad de trabajo, la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.

– Para confirmar o restituir este tipo de unidad de trabajo, utilice las llamadas de confirmación y devolución proporcionadas por el coordinador de unidad de trabajo externo.

  Si utiliza la llamada MQCMIT o MQBACK para confirmar o restituir la unidad de trabajo, la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.

2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC-Desconectar gestor de colas" en la página 693 para obtener más detalles.
3. Una aplicación sólo puede participar en una unidad de trabajo a la vez. La llamada MQBEGIN falla con el código de razón MQRC_UOW_IN_PROGRESS si ya existe una unidad de trabajo para la aplicación, independientemente del tipo de unidad de trabajo que sea.
4. La llamada MQBEGIN no es válida en un entorno de cliente MQI de MQ . Un intento de utilizar la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.
5. Cuando el gestor de colas actúa como coordinador de unidad de trabajo para unidades de trabajo globales, los gestores de recursos que pueden participar en la unidad de trabajo se definen en el archivo de configuración del gestor de colas.
6. En IBM i, los tres tipos de unidad de trabajo están soportados de la siguiente manera:
   - **Unidad de trabajo local coordinada por el gestor de colas** sólo se puede utilizar cuando no existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no se debe haber emitido para el trabajo.
   - **Unidad de trabajo global coordinada por el gestor de colas** no está soportada.
   - **Unidad de trabajo global coordinada externamente** sólo se puede utilizar cuando existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** debe haberse emitido para el trabajo. Si esto se ha hecho, las operaciones IBM i COMMIT y ROLLBACK se aplican a los recursos de MQ , así como a los recursos que pertenecen a otros gestores de recursos participantes.

Invocación en C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQBO     BeginOptions;  /* Options that control the action of MQBEGIN */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Options that control the action of MQBEGIN
 01  BEGINOPTIONS.
     COPY CMQBOV.
**   Completion code
```

```
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl BeginOptions  like MQBO;      /* Options that control the action of
                                     MQBEGIN */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn        As Long 'Connection handle'
Dim BeginOptions As MQBO 'Options that control the action of MQBEGIN'
Dim CompCode     As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

MQBUFMH - Convertir almacenamiento intermedio en descriptor de contexto de mensaje

La llamada de función MQBUFMH convierte un almacenamiento intermedio en un manejador de mensajes y es el inverso de la llamada MQMHBUF.

Esta llamada toma un descriptor de mensaje y las propiedades MQRFH2 del almacenamiento intermedio y las hace disponibles a través de un descriptor de mensaje. Las propiedades MQRFH2 de los datos del mensaje se eliminan, opcionalmente. Los campos *Encoding*, *CodedCharSetId*y *Format* del descriptor de mensaje se actualizan, si es necesario, para describir correctamente el contenido del almacenamiento intermedio después de que se hayan eliminado las propiedades.

Sintaxis

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de **Hconn** debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra convirtiendo un almacenamiento intermedio en un descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMQSG-entrada

Este es el descriptor de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

BufMsgHOpts
Tipo: MQBMHO-entrada

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios.

Para obtener más detalles, consulte "MQBMHO-Opciones de almacenamiento intermedio a manejador de mensajes" en la página 275.

MsgDesc
Tipo: MQMD - entrada/salida

La estructura *MsgDesc* contiene las propiedades del descriptor de mensaje y describe el contenido del área de almacenamiento intermedio.

En la salida de la llamada, las propiedades se eliminan opcionalmente del área de almacenamiento intermedio y, en este caso, el descriptor de mensaje se actualiza para describir correctamente el área de almacenamiento intermedio.

Los datos de esta estructura deben estar en el juego de caracteres y la codificación de la aplicación.

BufferLength
Tipo: MQLONG - entrada

BufferLength es la longitud del área de almacenamiento intermedio, en bytes.

Un *BufferLength* de cero bytes es válido e indica que el área de almacenamiento intermedio no contiene datos.

Almacenamiento intermedio
Tipo: MQBYTExBufferLongitud-entrada/salida

Estas son opciones que controlan la acción de MQBEGIN, tal como se describe en "MQBEGIN-Iniciar unidad de trabajo" en la página 639.

Buffer define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si **Buffer** contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BufferLength** es cero, no se hace referencia a **Buffer** ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DataLength
Tipo: MQLONG - salida

Longitud, en bytes, del almacenamiento intermedio que puede tener las propiedades eliminadas.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BMHO_ERROR
(2489, X'09B9') La estructura de opciones de almacenamiento intermedio a manejador de mensajes no es válida.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MD_ERROR
(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_RFH_ERROR
(2334, X'091E') La estructura MQRFH2 no es válida.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

Las llamadas MQBUFMH no se pueden interceptar mediante salidas de API: un almacenamiento intermedio se convierte en un manejador de mensajes en el espacio de aplicación; la llamada no llega al gestor de colas.

Invocación en C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;        /* Connection handle */
MQHMSG  Hmsg;         /* Message handle */
MQBMHO  BufMsgHOpts;  /* Options that control the action of MQBUFMH */
MQMD    MsgDesc;      /* Message descriptor */
MQLONG  BufferLength; /* Length in bytes of the Buffer area */
MQBYTE  Buffer[n];    /* Area to contain the message buffer */
MQLONG  DataLength;   /* Length of the output buffer */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                          BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Message handle
 01  HMSG          PIC S9(18) BINARY.
**   Options that control the action of MQBUFMH
 01  BUFMSGHOPTS.
     COPY CMQBMHOV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMD.
**   Length in bytes of the Buffer area
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the message buffer
 01  BUFFER        PIC X(n).
**   Length of the output buffer
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Hmsg         fixed bin(63); /* Message handle */
dcl BufMsgHOpts  like MQBMHO;   /* Options that control the action of
                                   MQBUFMH */
dcl MsgDesc      like MQMD;     /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer       char(n);       /* Area to contain the message buffer */
dcl DataLength   fixed bin(31); /* Length of the output buffer */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQBUFMH,(HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
                DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN        DS       F      Connection handle
HMSG         DS       D      Message handle
BUFMSGHOPTS  CMQBMHOA ,      Options that control the action of MQBUFMH
MSGDESC      CMQMDA   ,      Message descriptor
BUFFERLENGTH DS       F      Length in bytes of the BUFFER area
BUFFER       DS       CL(n)  Area to contain the properties
DATALENGTH   DS       F      Length of the output buffer
COMPCODE     DS       F      Completion code
REASON       DS       F      Reason code qualifying COMPCODE
```

MQCB-Gestionar devolución de llamada

La llamada MQCB registra una devolución de llamada para el descriptor de objeto especificado y controla la activación y los cambios en la devolución de llamada.

Una devolución de llamada es un fragmento de código (especificado como el nombre de una función que se puede enlazar dinámicamente o como puntero de función) al que llama IBM MQ cuando se producen determinados sucesos.

Para utilizar MQCB y MQCTL en un cliente, debe estar conectado a un servidor en el que el parámetro **SHARECNV** negociado del canal haya acordado un valor distinto de cero.

Los tipos de devolución de llamada que se pueden definir son:

Consumidor de mensajes

Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto.

Sólo se puede registrar una función de devolución de llamada para cada descriptor de contexto de objeto. Si se va a leer una sola cola con varios criterios de selección, la cola se debe abrir varias veces y se debe registrar una función de consumidor en cada descriptor de contexto.

Manejador de sucesos

Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de devolución de llamada.

Se llama a la función cuando se produce una condición de suceso, por ejemplo, un gestor de colas o una conexión que se detiene o se desactiva temporalmente.

La función no se invoca para condiciones específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INHIBITED; sin embargo, se llama si una función de devolución de llamada no finaliza normalmente.

Sintaxis

MQCB (*Hconn*, *Operación*, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS puede especificar el siguiente valor especial para que *MQHC_DEF_HCONN* utilice el descriptor de conexión asociado a esta unidad de ejecución.

Operación

Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una de las opciones siguientes. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

MQOP_REGISTRO

Defina la función de devolución de llamada para el descriptor de objeto especificado. Esta operación define la función que se va a llamar y los criterios de selección que se van a utilizar.

Si ya se ha definido una función de devolución de llamada para el descriptor de contexto de objeto, se sustituye la definición. Si se detecta un error al sustituir la devolución de llamada, se anula el registro de la función.

Si una devolución de llamada se registra en la misma función de devolución de llamada en la que se ha anulado el registro anteriormente, se trata como una operación de sustitución; no se invoca ninguna llamada inicial o final.

Puede utilizar MQOP_REGISTER con MQOP_SUSPEND o MQOP_RESUME.

MQOP_DEREGISTER

Detiene el consumo de mensajes para el descriptor de contexto de objeto y elimina el descriptor de contexto de los elegibles para una devolución de llamada.

Una devolución de llamada se anula automáticamente si se cierra el descriptor de contexto asociado.

Si se llama a MQOP_DEREGISTER desde dentro de un consumidor, y la devolución de llamada tiene definida una llamada de detención, se invoca tras la devolución del consumidor.

Si esta operación se emite en un *Hobj* sin ningún consumidor registrado, la llamada se devuelve con MQRC_CALLBACK_NOT_REGISTERED.

MQOP_SUSPENDER

Suspende el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

Mientras está suspendida, la función de consumidor continúa obteniendo las devoluciones de llamada de tipo de control.

MQOP_RESUME

Reanude el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

CallbackDesc

Tipo: MQCBD-entrada

Esta es una estructura que identifica la función de devolución de llamada que está registrando la aplicación y las opciones utilizadas al registrarla.

Consulte MQCBD para obtener detalles de la estructura.

El descriptor de devolución de llamada sólo es necesario para la opción MQOP_REGISTER; si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

Hobj

Tipo: MQHOBJ - entrada

Este descriptor de contexto representa el acceso que se ha establecido al objeto desde el que se va a consumir un mensaje. Es un descriptor de contexto que se ha devuelto desde una llamada MQOPEN o MQSUB anterior (en el parámetro **Hobj**).

Hobj no es necesario al definir una rutina de manejador de sucesos (MQCBT_EVENT_HANDLER) y debe especificarse como MQHO_NONE.

Si se ha devuelto *Hobj* desde una llamada MQOPEN, la cola debe haberse abierto con una o varias de las opciones siguientes:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Tipo: MQMD-entrada

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado.

El parámetro **MsgDesc** define los atributos de los mensajes que necesita el consumidor y la versión del MQMD que se va a pasar al consumidor de mensajes.

MsgId, *CorrelId*, *GroupId*, *MsgSeqNumber*y *Offset* en MQMD se utilizan para la selección de mensajes, en función de las opciones especificadas en el parámetro **GetMsgOpts** .

Encoding y *CodedCharSetId* se utilizan para la conversión de mensajes si especifica la opción MQGMO_CONVERT.

Consulte MQMD para obtener más detalles.

MsgDesc se utiliza para MQOP_REGISTER y si necesita valores distintos del valor predeterminado para cualquier campo. *MsgDesc* no se utiliza para un manejador de sucesos.

Si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

Tenga en cuenta que si se registran varios consumidores en la misma cola con selectores solapados, el consumidor elegido para cada mensaje no está definido.

GetMsgOpts

Tipo: MQGMO-entrada

El parámetro **GetMsgOpts** controla cómo obtiene los mensajes el consumidor de mensajes. Todas las opciones de este parámetro tienen significados tal como se describe en "MQGMO-Opciones de obtención de mensajes" en la página 367, cuando se utilizan en una llamada MQGET, excepto:

MQGMO_SET_SIGNAL

Esta opción no está permitida.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_ *

El orden de los mensajes entregados a un consumidor de navegación viene determinado por las combinaciones de estas opciones. Las combinaciones significativas son:

MQGMO_BROWSE_FIRST

El primer mensaje de la cola se entrega repetidamente al consumidor. Esto es útil cuando el consumidor consume de forma destructiva el mensaje en la devolución de llamada. Utilice esta opción con cuidado.

MQGMO_BROWSE_NEXT

Al consumidor se le asigna cada mensaje de la cola, desde la posición actual del cursor hasta que se alcanza el final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT
El cursor se restablece al inicio de la cola. A continuación, se proporciona al consumidor cada mensaje hasta que el cursor llega al final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_MARK_ *
A partir del principio de la cola, al consumidor se le proporciona el primer mensaje no marcado en la cola, que a continuación se marca para este consumidor. Esta combinación garantiza que el consumidor pueda recibir nuevos mensajes añadidos detrás del punto de cursor actual.

MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
A partir de la posición del cursor, al consumidor se le proporciona el siguiente mensaje no marcado en la cola, que a continuación se marca para este consumidor. Utilice esta combinación con cuidado porque los mensajes se pueden añadir a la cola detrás de la posición actual del cursor.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
Esta combinación no está permitida. Si se utiliza, la llamada devuelve MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT y WaitInterval
Estas opciones controlan cómo se invoca al consumidor.

MQGMO_NO_WAIT
Nunca se llama al consumidor con MQRC_NO_MSG_AVAILABLE. Sólo se llama al consumidor para mensajes y sucesos.

MQGMO_WAIT con un WaitInterval cero
El código MQRC_NO_MSG_AVAILABLE se pasa al consumidor cuando no hay mensajes disponibles y el consumidor se ha iniciado o se ha entregado al menos un mensaje desde el último código de razón "sin mensajes".

Esto impide que el consumidor sondee en un bucle ocupado cuando se especifica un intervalo de espera cero.

MQGMO_WAIT y un WaitInterval positivo
Se llama al consumidor después del intervalo de espera especificado con el código de razón MQRC_NO_MSG_AVAILABLE. Esta llamada se realiza independientemente de si se ha entregado algún mensaje al consumidor. Esto permite al usuario realizar un proceso de latido o de tipo de proceso por lotes.

MQGMO_WAIT y WaitInterval de MQWI_UNLIMITED
Especifica una espera infinita antes de devolver MQRC_NO_MSG_AVAILABLE. Nunca se llama al consumidor con MQRC_NO_MSG_AVAILABLE.

GetMsgOpts sólo se utiliza para MQOP_REGISTER y si necesita valores distintos del valor predeterminado para cualquier campo. *GetMsgOpts* no se utiliza para un manejador de sucesos.

Si los *GetMsgOpts* no son necesarios, la dirección de parámetro pasada puede ser nula. El uso de este parámetro es el mismo que especificar MQGMO_DEFAULT junto con MQGMO_FAIL_IF_QUIESCING.

Si se proporciona un descriptor de contexto de propiedades de mensaje en la estructura MQGMO, se proporciona una copia en la estructura MQGMO que se pasa a la devolución de llamada del consumidor. Al volver de la llamada MQCB, la aplicación puede suprimir el descriptor de contexto de propiedades de mensaje.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón de la lista siguiente son los que el gestor de colas puede devolver para el parámetro **Reason** .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Campo de tipo de devolución de llamada incorrecto.

MQRC_CALLBACK_NOT_REGISTERED
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada.

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Se debe especificar *CallbackFunction* o *CallbackName* , pero no ambos.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Campo de tipo de devolución de llamada incorrecto.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B4') Campo *MaxMsgLength* incorrecto.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') No se ha podido encontrar el punto de entrada de función especificado en el módulo.

MQRC_MODULE_INVALID
(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto; no de 32 bits, 64 bits o una biblioteca de enlace dinámico válida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPERATION_ERROR
(2206, X'89E') Código de operación incorrecto en llamada de API.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. MQCB se utiliza para definir la acción que se va a invocar para cada mensaje, que coincide con los criterios especificados, disponibles en la cola. Cuando se procesa la acción, el mensaje se elimina de la cola y se pasa al consumidor de mensajes definido, o se proporciona una señal de mensaje, que se utiliza para recuperar el mensaje.
2. MQCB se puede utilizar para definir rutinas de devolución de llamada antes de iniciar el consumo con MQCTL o se puede utilizar desde dentro de una rutina de devolución de llamada.
3. Para utilizar MQCB desde fuera de una rutina de devolución de llamada, primero debe suspender el consumo de mensajes utilizando MQCTL y reanudar el consumo posteriormente.
4. MQCB no está soportado en el adaptador IMS .

Secuencia de devolución de llamada de consumidor de mensajes

Puede configurar un consumidor para invocar la devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor esté registrado por primera vez,
- cuando se inicia la conexión,
- cuando se detiene la conexión y
- cuando se anula el registro del consumidor, ya sea explícita o implícitamente mediante un MQCLOSE.

Tabla 541. Definiciones de verbo MQCTL

Verbo	Significado
MQCTL (START)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL (ESPERAR)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación de consumidor son las siguientes:

REGISTER
Es siempre el primer tipo de invocación de la devolución de llamada.

Siempre se llama en la misma hebra, como la llamada MQCB (REGISTER).

START
Siempre se llama de forma síncrona con el verbo MQCTL (START).

- Todas las devoluciones de llamada START se completan antes de que se devuelva el verbo MQCTL (START).

Está en la misma hebra que la entrega de mensajes si se solicita THREAD_AFFINITY.

La llamada con inicio no se garantiza si, por ejemplo, una devolución de llamada anterior emite MQCTL (STOP) durante MQCTL (START).

STOP
No se entregan más mensajes o sucesos después de esta llamada hasta que se reinicia la conexión.

Se garantiza un STOP si la aplicación se ha llamado anteriormente para START, o un mensaje, o un suceso.

DEREGISTER
Es siempre el último tipo de invocación de la devolución de llamada.

Asegúrese de que la aplicación realiza la inicialización y limpieza basadas en hebras en las devoluciones de llamada START y STOP. Puede realizar una inicialización y limpieza no basadas en hebras con devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida y la disponibilidad del hilo aparte de lo que se indica. Por ejemplo, no confíe en que una hebra permanezca activa más allá de la última llamada a DEREGISTER. De forma similar, cuando haya elegido no utilizar THREAD_AFFINITY, no presuponga que la hebra existe siempre que se inicie la conexión.

Si la aplicación tiene requisitos específicos para las características de hebra, siempre puede crear una hebra en consecuencia y, a continuación, utilizar MQCTL (WAIT). Esto tiene el efecto de 'donar' la hebra a IBM MQ para la entrega de mensajes asíncronos.

Uso de conexión de consumidor de mensajes

Puede configurar un consumidor para invocar la devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor esté registrado por primera vez,
- cuando se inicia la conexión,
- cuando se detiene la conexión y
- cuando se anula el registro del consumidor, ya sea explícita o implícitamente mediante un MQCLOSE.

Tabla 542. Definiciones de verbo MQCTL

Verbo	Significado
MQCTL (START)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL (ESPERAR)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación de consumidor son las siguientes:

REGISTER
Es siempre el primer tipo de invocación de la devolución de llamada.

Siempre se llama en la misma hebra, como la llamada MQCB (REGISTER).

START
Siempre se llama de forma síncrona con el verbo MQCTL (START).

- Todas las devoluciones de llamada START se completan antes de que se devuelva el verbo MQCTL (START).

Está en la misma hebra que la entrega de mensajes si se solicita THREAD_AFFINITY.

La llamada con inicio no se garantiza si, por ejemplo, una devolución de llamada anterior emite MQCTL (STOP) durante MQCTL (START).

STOP
No se entregan más mensajes o sucesos después de esta llamada hasta que se reinicia la conexión.

Se garantiza un STOP si la aplicación se ha llamado anteriormente para START, o un mensaje, o un suceso.

DEREGISTER
Es siempre el último tipo de invocación de la devolución de llamada.

Asegúrese de que la aplicación realiza la inicialización y limpieza basadas en hebras en las devoluciones de llamada START y STOP. Puede realizar una inicialización y limpieza no basadas en hebras con devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida y la disponibilidad del hilo aparte de lo que se indica. Por ejemplo, no confíe en que una hebra permanezca activa más allá de la última llamada a DEREGISTER. De forma similar, cuando haya elegido no utilizar THREAD_AFFINITY, no presuponga que la hebra existe siempre que se inicie la conexión.

Si la aplicación tiene requisitos específicos para las características de hebra, siempre puede crear una hebra en consecuencia y, a continuación, utilizar MQCTL (WAIT). Esto tiene el efecto de 'donar' la hebra a IBM MQ para la entrega de mensajes asíncronos.

Invocación en C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
GetMsgOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQLONG   Operation;     /* Operation being processed */
MQCBD    CallbackDesc;  /* Callback descriptor */
MQHOBJ   HObj           /* Object handle */
MQMD     MsgDesc        /* Message descriptor attributes */
MQGMO    GetMsgOpts     /* Message options */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,
            GETMSGOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Operation
 01  OPERATION PIC S9(9) BINARY.
**   Callback Descriptior
 01  CBDESC.
     COPY CMQCBDV.
01  HOBJ     PIC S9(9) BINARY.
**   Message Descriptior
 01  MSGDESC.
     COPY CMQMDV.
**   Get Message Options
 01  GETMSGOPTS.
     COPY CMQGMOV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
          CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Operation    fixed bin(31); /* Operation */
dcl CallbackDesc like MQCBD;    /* Callback Descriptor */
dcl Hobj         fixed bin(31); /* Object Handle */
dcl MsgDesc      like MQMD;     /* Message Descriptor */
```

```
dcl GetMsgOpts   like MQGMO;    /* Get Message Options */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

MQCB_FUNCTION-Función de devolución de llamada

La llamada a la función MQCB_FUNCTION es la función de devolución de llamada para el manejo de sucesos y el consumo de mensajes asíncronos.

La definición de llamada MQCB_FUNCTION se proporciona únicamente para describir los parámetros que se pasan a la función de devolución de llamada. El gestor de colas no proporciona ningún punto de entrada denominado MQCB_FUNCTION.

La especificación de la función real que se va a llamar es una entrada a la llamada MQCB y se pasa a través de la estructura MQCBD .

Sintaxis

`MQCB_FUNCTION` (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Contexto*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior. En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para Hconn:

MQHC_DEF_CONN

Manejador de conexión predeterminado.

MsgDesc

Tipo: MQMD-entrada

Esta estructura describe los atributos del mensaje recuperado.

Consulte “MQMD - Descriptor de mensaje” en la página 421 para obtener los detalles.

La versión de MQMD pasada es la misma versión que la pasada en la llamada MQCB que ha definido la función de consumidor.

La dirección del MQMD se pasa como caracteres nulos si se ha utilizado un MQGMO de la versión 4 para solicitar que se devuelva un manejador de mensajes en lugar de un MQMD.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

GetMsgOpts

Tipo: MQGMO-entrada

Opciones utilizadas para controlar las acciones del consumidor de mensajes. Este parámetro también contiene información adicional sobre el mensaje devuelto.

Consulte MQGMO para obtener más detalles.

La versión de MQGMO pasada es la última versión soportada.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Almacenamiento intermedio

Tipo: MQBYTExBufferLongitud-entrada

Es el área que contiene los datos del mensaje.

Si no hay ningún mensaje disponible para esta llamada, o si el mensaje no contiene datos de mensaje, la dirección del *Buffer* se pasa como nulos.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Contexto
Tipo: MQCBC-entrada/salida

Esta estructura proporciona información de contexto a las funciones de devolución de llamada. Consulte "MQCBC-Contexto de devolución de llamada" en la página 279 para obtener los detalles.

Notas de uso

1. Tenga en cuenta que si las rutinas de devolución de llamada utilizan servicios que podrían retrasar o bloquear la hebra, por ejemplo, MQGET con espera, podría retrasar la asignación de otras devoluciones de llamada.
2. No se establece automáticamente una unidad de trabajo separada para cada invocación de una rutina de devolución de llamada, por lo que las rutinas pueden emitir una llamada de confirmación, o aplazar la confirmación, hasta que se haya procesado un lote lógico de trabajo. Cuando se confirma el lote de trabajo, confirma los mensajes para todas las funciones de devolución de llamada que se han invocado desde el último punto de sincronización.
3. Los programas invocados por CICS LINK o CICS START recuperan parámetros utilizando servicios CICS a través de objetos con nombre conocidos como contenedores de canal. Los nombres de contenedor son los mismos que los nombres de parámetro. Para obtener más información, consulte la documentación de CICS.
4. Las rutinas de devolución de llamada pueden emitir una llamada MQDISC, pero no para su propia conexión. Por ejemplo, si una rutina de devolución de llamada ha creado una conexión, también puede desconectar la conexión.
5. Una rutina de devolución de llamada no debe, en general, basarse en que se invoque desde la misma hebra cada vez. Si es necesario, utilice MQCTLO_THREAD_AFFINITY cuando se inicie la conexión.
6. Cuando una rutina de devolución de llamada recibe un código de razón distinto de cero, debe realizar la acción adecuada.
7. MQCB_FUNCTION no está soportado en el adaptador IMS .

MQCLOSE-Cerrar objeto

La llamada MQCLOSE renuncia al acceso a un objeto y es la inversa de las llamadas MQOPEN y MQSUB.

Sintaxis

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS puede omitir la llamada MQCONN y especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj
Tipo: MQHOBJ - entrada/salida

Este descriptor de contexto representa el objeto que se está cerrando. El objeto puede ser de cualquier tipo. El valor de *Hobj* ha sido devuelto por una llamada MQOPEN anterior.

Al finalizar correctamente la llamada, el gestor de colas establece este parámetro en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

MQHO_UNUSABLE_HOBJ
Descriptor de objeto inutilizable.

En z/OS, *Hobj* se establece en un valor que no está definido.

Opciones
Tipo: MQLONG - entrada

Este parámetro controla cómo se cierra el objeto.

Sólo las colas dinámicas permanentes y las suscripciones se pueden cerrar de más de una forma, porque se deben retener o suprimir; estas son colas con el atributo **DefinitionType** que tiene el valor MQQDT_PERMANENT_DYNAMIC (consulte el atributo **DefinitionType** descrito en "Atributos para colas" en la página 848). Las opciones de cierre se resumen en este tema.

Las suscripciones duraderas se pueden conservar o eliminar; estas se crean utilizando la llamada MQSUB con la opción MQSO_DURABLE.

Al cerrar el descriptor de contexto en un destino gestionado (es decir, el parámetro **Hobj** devuelto en una llamada MQSUB que ha utilizado la opción MQSO_MANAGED), el gestor de colas limpia las publicaciones que no se han recuperado cuando también se ha eliminado la suscripción asociada. La suscripción se elimina utilizando la opción MQCO_REMOVE_SUB en el parámetro **Hsub** devuelto en una llamada MQSUB. Tenga en cuenta que MQCO_REMOVE_SUB es el comportamiento predeterminado en MQCLOSE para una suscripción no duradera.

Al cerrar un descriptor de contexto en un destino no gestionado, es responsable de limpiar la cola donde se envían las publicaciones. Cierre primero la suscripción utilizando MQCO_REMOVE_SUB y, a continuación, procese los mensajes fuera de la cola hasta que no quede ninguno.

Sólo debe especificar una opción entre las siguientes:

Opciones de cola dinámica: estas opciones controlan cómo se cierran las colas dinámicas permanentes.

MQCO_DELETE
La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay mensajes en la cola y no hay solicitudes de obtención o colocación no confirmadas pendientes para la cola (para la tarea actual o para cualquier otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *Hobj*. En este caso, se depuran todos los mensajes de la cola.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón MQRC_OPTION_NOT_VALID_FOR_TYPE, y el objeto no se suprime.

En z/OS, si la cola es una cola dinámica que se ha suprimido lógicamente, y este es el último descriptor de contexto para ella, la cola se suprime físicamente. Para conocer detalles, consulte "Notas de uso" en la página 663.

MQCO_DELETE_PURGE
La cola se suprime y los mensajes que contiene se depuran, si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay solicitudes get o put no confirmadas pendientes para la cola (ni para la tarea actual ni para ninguna otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *Hobj*.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón MQRC_OPTION_NOT_VALID_FOR_TYPE, y el objeto no se suprime.

Tabla 543. Opciones de cierre para distintos tipos de objeto

Tipo de objeto o cola	**MQCO_NONE**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Objeto que no es una cola	Retenido	No válido	No válido
Cola predefinida	Retenido	No válido	No válido
cola dinámica permanente	Retenido	Suprimido si está vacío y no hay actualizaciones pendientes	Mensajes suprimidos; cola suprimida si no hay actualizaciones pendientes
Cola dinámica temporal (llamada emitida por el creador de la cola)	Suprimida	Suprimida	Suprimida
Cola dinámica temporal (llamada no emitida por el creador de la cola)	Retenido	No válido	No válido
Lista de distribución	Retenido	No válido	No válido
Destino de suscripción gestionada	Retenido	No válido	No válido
Lista de distribución (se ha eliminado la suscripción)	Mensajes suprimidos; cola suprimida	No válido	No válido

Opciones de cierre de suscripción: Estas opciones controlan si se eliminan las suscripciones duraderas cuando se cierra el descriptor de contexto y si se limpian las publicaciones que todavía están a la espera de ser leídas por la aplicación. Estas opciones sólo son válidas para su uso con un descriptor de objeto devuelto en el parámetro **Hsub** de una llamada MQSUB.

MQCO_KEEP_SUB
El descriptor de contexto de la suscripción se cierra, pero la suscripción realizada se mantiene. Las publicaciones se siguen enviando al destino especificado en la suscripción. Esta opción sólo es válida si la suscripción se ha realizado con la opción MQSO_DURABLE.

MQCO_KEEP_SUB es el valor predeterminado si la suscripción es duradera

MQCO_REMOVE_SUB
La suscripción se elimina y se cierra el descriptor de contexto de la suscripción.

El parámetro **Hobj** de la llamada MQSUB no se invalida mediante el cierre del parámetro **Hsub** y puede seguir utilizándose para MQGET o MQCB para recibir las publicaciones restantes. Cuando el parámetro **Hobj** de la llamada MQSUB también se cierra, si era un destino gestionado se eliminan las publicaciones no recuperadas.

MQCO_REMOVE_SUB es el valor predeterminado si la suscripción no es duradera.

La finalización satisfactoria de MQCO_REMOVE_SUB no significa que la acción se haya completado. Para comprobar que esta llamada se ha completado, consulte el paso DELETE SUB en Comprobación de que los mandatos asíncronos para redes distribuidas han finalizado.

Estas opciones de cierre de suscripción se resumen en las tablas siguientes.

Tabla 544. Opciones para cerrar un descriptor de contexto de suscripción duradera pero conservar la suscripción	
Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	MQCO_KEEP_SUB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto MQSO_MANAGED	MQCO_KEEP_SUB
Eliminar publicaciones en un descriptor de contexto MQSO_MANAGED	Acción no permitida

Para anular la suscripción, ya sea cerrando un descriptor de contexto de suscripción duradera y cancelándola o cerrando un descriptor de contexto de suscripción no duradera, utilice las siguientes opciones de cierre de suscripción:

Tabla 545. Opciones para anular la suscripción	
Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	MQCO_REMOVE_SUB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto MQSO_MANAGED	MQCO_REMOVE_SUB

Opciones de lectura anticipada: Las opciones siguientes controlan lo que sucede con los mensajes no persistentes que se han enviado al cliente antes de que una aplicación los solicitara y que todavía no han sido consumidos por la aplicación. Estos mensajes se almacenan en el almacenamiento intermedio de lectura anticipada del cliente a la espera de ser solicitados por la aplicación y se pueden descartar o consumir de la cola antes de que se complete MQCLOSE.

MQCO_INMEDIATO
El objeto se cierra inmediatamente y los mensajes que se han enviado al cliente antes de que una aplicación los solicitara se descartan y no están disponibles para que los consuma ninguna aplicación. Éste es el valor predeterminado.

MQCO_QUIESCE
Se realiza una solicitud para cerrar el objeto, pero si alguno de los mensajes que se han enviado al cliente antes de que una aplicación los solicitara, sigue residiendo en el almacenamiento intermedio de lectura anticipada del cliente, la llamada MQCLOSE devuelve un aviso de MQRC_READ_AHEAD_MSGS y el descriptor de contexto del objeto sigue siendo válido.

A continuación, la aplicación puede seguir utilizando el descriptor de objeto para recuperar mensajes hasta que no haya más disponibles y, a continuación, vuelva a cerrar el objeto. No se envían más mensajes al cliente antes de que una aplicación los solicite, la lectura anticipada ahora está desactivada.

Se recomienda a las aplicaciones que utilicen MQCO_QUIESCE en lugar de intentar alcanzar un punto en el que no haya más mensajes en el almacenamiento intermedio de lectura anticipada del cliente, porque podría llegar un mensaje entre la última llamada MQGET y el siguiente MQCLOSE que se descartaría si se utilizara MQCO_IMMEDIATE.

Si se emite un MQCLOSE con MQCO_QUIESCE desde una función de devolución de llamada asíncrona, se aplica el mismo comportamiento de lectura anticipada de mensajes. Si se devuelve el aviso MQRC_READ_AHEAD_MSGS, se llama a la función de devolución de llamada al menos

una vez más. Cuando el último mensaje restante que se ha leído con anticipación se ha pasado a la función de devolución de llamada, el campo ConsumerFlags de MQCBC se establece en MQCBCF_READA_BUFFER_EMPTY.

Opción predeterminada: Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

MQCO_NONE
No es necesario ningún proceso de cierre opcional.

Debe especificarse para:

- Objetos que no son colas
- Colas predefinidas
- Colas dinámicas temporales (pero sólo en los casos en los que *Hobj* no es el manejador devuelto por la llamada MQOPEN que ha creado la cola).
- Listas de distribución

En todos los casos anteriores, el objeto se conserva y no se suprime.

Si se especifica esta opción para una cola dinámica temporal:

- La cola se suprime, si la ha creado la llamada MQOPEN que ha devuelto *Hobj* ; los mensajes que están en la cola se depuran.
- En todos los demás casos, la cola (y los mensajes que contiene) se retienen.

Si se especifica esta opción para una cola dinámica permanente, la cola se conserva y no se suprime.

En z/OS, si la cola es una cola dinámica que se ha suprimido lógicamente, y este es el último descriptor de contexto para ella, la cola se suprime físicamente. Para conocer detalles, consulte "Notas de uso" en la página 663.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón listados son los que el gestor de colas puede devolver para el parámetro **Reason** .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx ') El cliente tiene mensajes de lectura anticipada que la aplicación aún no ha consumido.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') En una llamada MQOPEN o MQCLOSE: opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_NOT_EMPTY
(2055, X'807 ') La cola contiene uno o más mensajes o solicitudes put u get no confirmadas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Cuando una aplicación emite la llamada MQDISC, o finaliza de forma normal o anómala, los objetos abiertos por la aplicación y que siguen abiertos se cierran automáticamente con la opción MQCO_NONE.
2. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola*:
   - Si las operaciones en la cola se realizan como parte de una unidad de trabajo, la cola se puede cerrar antes o después de que se produzca el punto de sincronización sin afectar al resultado del punto de sincronización. Si se desencadena la cola, la realización de una retrotracción antes de cerrar la cola puede hacer que se emita un mensaje desencadenante. Para obtener más información sobre los mensajes desencadenantes, consulte Propiedades de los mensajes desencadenantes.
   - Si la cola se ha abierto con la opción MQOO_BROWSE, el cursor para examinar se destruye. Si la cola se vuelve a abrir con la opción MQOO_BROWSE, se crea un nuevo cursor para examinar (consulte MQOO_BROWSE).
   - Si un mensaje está bloqueado actualmente para este descriptor de contexto en el momento de la llamada MQCLOSE, el bloqueo se libera (consulte MQGMO_LOCK).
   - En z/OS, si hay una solicitud MQGET con la opción MQGMO_SET_SIGNAL pendiente en el descriptor de contexto de cola que se está cerrando, la solicitud se cancela (consulte MQGMO_SET_SIGNAL). Las solicitudes de señal para la misma cola pero alojadas en distintos manejadores (*Hobj*) no se ven afectadas (a menos que se suprima una cola dinámica, en cuyo caso también se cancelan).
3. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola dinámica* (permanente o temporal):
   - Para una cola dinámica, puede especificar las opciones MQCO_DELETE y MQCO_DELETE_PURGE independientemente de las opciones especificadas en la llamada MQOPEN correspondiente.
   - Cuando se suprime una cola dinámica, se cancelan todas las llamadas MQGET con la opción MQGMO_WAIT que están pendientes en la cola y se devuelve el código de razón MQRC_Q_DELETED. Consulte MQGMO_WAIT.

     Aunque las aplicaciones no pueden acceder a una cola suprimida, la cola no se elimina del sistema y los recursos asociados no se liberan, hasta que se hayan cerrado todos los descriptores de contexto que hacen referencia a la cola y todas las unidades de trabajo que afectan a la cola se hayan confirmado o restituido.

En z/OS, una cola que se ha suprimido lógicamente pero que todavía no se ha eliminado del sistema impide la creación de una cola nueva con el mismo nombre que la cola suprimida; la llamada MQOPEN falla con el código de razón MQRC_NAME_IN_USE en este caso. Además, una cola de este tipo se puede seguir visualizando utilizando mandatos MQSC, aunque las aplicaciones no puedan acceder a ella.

- Cuando se suprime una cola dinámica permanente, si el descriptor de contexto *Hobj* especificado en la llamada MQCLOSE no es el que ha devuelto la llamada MQOPEN que ha creado la cola, se comprueba que el identificador de usuario que se ha utilizado para validar la llamada MQOPEN esté autorizado para suprimir la cola. Si se ha especificado la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN, el identificador de usuario seleccionado es *AlternateUserId*.

  Esta comprobación no se realiza si:

  – El descriptor de contexto especificado es el devuelto por la llamada MQOPEN que ha creado la cola.

  – La cola que se está suprimiendo es una cola dinámica temporal.

- Cuando se cierra una cola dinámica temporal, si el descriptor de contexto *Hobj* especificado en la llamada MQCLOSE es el que ha devuelto la llamada MQOPEN que ha creado la cola, se suprime la cola. Esto ocurre independientemente de las opciones de cierre especificadas en la llamada MQCLOSE. Si hay mensajes en la cola, se descartan; no se generan mensajes de informe.

  Si hay unidades de trabajo no confirmadas que afectan a la cola, la cola y sus mensajes se siguen suprimiendo, pero las unidades de trabajo no fallan. Sin embargo, como se ha descrito anteriormente, los recursos asociados con las unidades de trabajo no se liberan hasta que se haya confirmado o restituido cada una de las unidades de trabajo.

4. Los puntos siguientes se aplican si el objeto que se está cerrando es una *lista de distribución*:

   - La única opción de cierre válida para una lista de distribución es MQCO_NONE; la llamada falla con el código de razón MQRC_OPTIONS_ERROR o MQRC_OPTION_NOT_VALID_FOR_TYPE si se especifica alguna otra opción.
   - Cuando se cierra una lista de distribución, los códigos de terminación y los códigos de razón individuales no se devuelven para las colas de la lista; sólo los parámetros **CompCode** y **Reason** de la llamada están disponibles para fines de diagnóstico.

     Si se produce una anomalía al cerrar una de las colas, el gestor de colas continúa el proceso e intenta cerrar las colas restantes de la lista de distribución. Los parámetros **CompCode** y **Reason** de la llamada se establecen para devolver información que describe la anomalía. Es posible que el código de finalización sea MQCC_FAILED, aunque la mayoría de las colas se hayan cerrado correctamente. La cola que ha encontrado el error no está identificada.

     Si hay una anomalía en más de una cola, no se define qué anomalía se notifica en los parámetros **CompCode** y **Reason** .

Invocación en C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQHOBJ   Hobj;      /* Object handle */
MQLONG   Options;   /* Options that control the action of MQCLOSE */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Object handle
 01  HOBJ      PIC S9(9) BINARY.
**   Options that control the action of MQCLOSE
 01  OPTIONS   PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl Hobj      fixed bin(31);  /* Object handle */
dcl Options   fixed bin(31);  /* Options that control the action of
                                 MQCLOSE */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
HOBJ      DS  F  Object handle
OPTIONS   DS  F  Options that control the action of MQCLOSE
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim Hobj     As Long 'Object handle'
Dim Options  As Long 'Options that control the action of MQCLOSE'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCMIT-Confirmar cambios

La llamada MQCMIT indica al gestor de colas que la aplicación ha alcanzado un punto de sincronización y que todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización se deben hacer permanentes.

Los mensajes colocados como parte de una unidad de trabajo se ponen a disposición de otras aplicaciones; los mensajes recuperados como parte de una unidad de trabajo se suprimen.

- z/OS En z/OS, la llamada sólo la utilizan los programas por lotes (incluidos los programas DL/I por lotes IMS).

Sintaxis

MQCMIT (*Hconn*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Los códigos de razón listados son los que el gestor de colas puede devolver para el parámetro **Reason** .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_OUTCOME_PENDING
(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT o MQCMIT se ha interrumpido y el proceso de reconexión no puede restablecer un resultado definido.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Después de la reconexión, se ha producido un error al restablecer los descriptores de contexto para una conexión reconectable.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Utilice esta llamada sólo cuando el propio gestor de colas coordine la unidad de trabajo. Este puede ser:
   - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
   - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de IBM MQ .

   Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte “MQBEGIN-Iniciar unidad de trabajo” en la página 639.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de confirmación adecuada en lugar de MQCMIT. El entorno también puede dar soporte a una confirmación implícita causada por la terminación normal de la aplicación.
   - En z/OS, utilice las llamadas siguientes:
     - Los programas por lotes (incluidos los programas DL/I por lotes IMS) pueden utilizar la llamada MQCMIT si la unidad de trabajo sólo afecta a los recursos de IBM MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de IBM MQ como a los recursos que pertenecen a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRCMIT proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRCMIT confirma los cambios en

los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.

  - Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT` para confirmar la unidad de trabajo de forma explícita. De forma alternativa, la finalización de la transacción da como resultado una confirmación implícita de la unidad de trabajo. La llamada MQCMIT no se puede utilizar para aplicaciones CICS .
  - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como GU y CHKP para confirmar la unidad de trabajo. La llamada MQCMIT no se puede utilizar para aplicaciones IMS (que no sean programas DL/I por lotes).
- En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.

3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso de MQDISC para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   Cuando se confirma una unidad de trabajo, el gestor de colas conserva la información de grupo y segmento, y la aplicación puede continuar colocando u obteniendo mensajes en el grupo de mensajes o mensaje lógico actual.

   La retención de la información de grupo y segmento cuando se confirma una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo. El uso de varias unidades de trabajo es ventajoso si el gestor de colas local sólo tiene almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema. Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte MQPMO_LOGICAL_ORDER y MQGMO_LOGICAL_ORDER.

   Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:
5. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión; todas las llamadas IBM MQ que afectan a una unidad de trabajo determinada deben realizarse utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro **Hconn** descrito en MQCONN para obtener información sobre el ámbito de los manejadores de conexión.
6. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
7. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o de devolución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para evitar esto, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

8. Windows UNIX En los sistemas UNIX y Windows , si el parámetro **Reason** es MQRC_CONNECTION_BROKEN (con un *CompCode* de MQCC_FAILED) o MQRC_UNEXPECTED_ERROR, es posible que la unidad de trabajo se haya confirmado correctamente.

Invocación en C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCONN-Conectar gestor de colas

La llamada MQCONN conecta un programa de aplicación a un gestor de colas.

Proporciona un descriptor de conexión del gestor de colas, que la aplicación utiliza en las llamadas de cola de mensajes posteriores.

- En z/OS, las aplicaciones CICS no tienen que emitir esta llamada. Estas aplicaciones se conectan automáticamente al gestor de colas al que está conectado el sistema CICS . Sin embargo, las llamadas MQCONN y MQDISC se siguen aceptando de las aplicaciones CICS .
- En IBM i, las aplicaciones deben utilizar la llamada MQCONN o MQCONNX para conectarse al gestor de colas y la llamada MQDISC para desconectarse del gestor de colas.

No se puede realizar una conexión de cliente en una instalación sólo de servidor, y no se puede realizar una conexión local en una instalación sólo de cliente.

Sintaxis

MQCONN (*QMgrName*, *Hconn*, *CompCode*, *Razón*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Es el nombre del gestor de colas al que la aplicación desea conectarse. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Se puede utilizar un carácter nulo para indicar el final de datos significativos en el nombre; el nulo y cualquier carácter que lo siga se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS, los nombres que empiezan o terminan con un subrayado no pueden ser procesados por las operaciones y los paneles de control. Por esta razón, evite este tipo de nombres.
- En IBM i, escriba los nombres que contengan caracteres en minúsculas, barra inclinada o porcentaje entre comillas cuando se especifiquen en los mandatos. No especifique estas comillas en el parámetro **QMgrName** .

Si el nombre consta por completo de espacios en blanco, se utiliza el nombre del gestor de colas *por omisión* . Sin embargo, tenga en cuenta el uso de nombres de gestor de colas en blanco que se describen en la sección sobre aplicaciones IBM MQ MQI client .

El nombre especificado para *QMgrName* debe ser el nombre de un gestor de colas *conectable* o, si se están utilizando grupos de gestores de colas, el nombre del grupo de gestores de colas.

En z/OS, los gestores de colas a los que es posible conectarse están determinados por el entorno:

- Para CICS, solo puede utilizar el gestor de colas al que está conectado el sistema CICS . Todavía debe especificarse el parámetro **QMgrName** , pero su valor se ignora; los caracteres en blanco son una opción adecuada.

- Para IMS, sólo se pueden conectar los gestores de colas listados en la tabla de definiciones de subsistema (CSQQDEFV), *y* listados en la tabla SSM en IMS(consulte la nota de uso 6).
- Para el proceso por lotes z/OS y TSO, sólo se pueden conectar los gestores de colas que residen en el mismo sistema que la aplicación (consulte la nota de uso 6).

Grupos de compartición de colas:En sistemas donde existen varios gestores de colas y están configurados para formar un grupo de compartición de colas, el nombre del grupo de compartición de colas se puede especificar para *QMgrName* en lugar del nombre de un gestor de colas. Esto permite a la aplicación conectarse a *cualquier* gestor de colas que esté disponible en el grupo de compartición de colas y que esté en la misma imagen de z/OS que la aplicación. El sistema también se puede configurar para que el uso de un *QMgrName* en blanco se conecte al grupo de compartición de colas en lugar de al gestor de colas predeterminado.

Si *QMgrName* especifica el nombre del grupo de compartición de colas, pero también hay un gestor de colas con ese nombre en el sistema, se establece una conexión con el último en lugar del primero. Sólo si la conexión falla, se intenta la conexión con uno de los gestores de colas del grupo de compartición de colas.

Si la conexión es satisfactoria, puede utilizar el descriptor de contexto devuelto por la llamada MQCONN o MQCONNX para acceder a *todos* los recursos (compartidos y no compartidos) que pertenecen al gestor de colas al que se ha realizado la conexión. El acceso a estos recursos está sujeto a los controles de autorización típicos.

Si la aplicación emite dos llamadas MQCONN o MQCONNX para establecer conexiones simultáneas, y una o ambas llamadas especifica el nombre del grupo de compartición de colas, la segunda llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED cuando se conecta al mismo gestor de colas que la primera llamada.

Los grupos de compartición de colas solo se admiten en z/OS. La conexión a un grupo de compartición de colas solo está soportada en los entornos por lotes, por lotes RRS, CICSy TSO. Para CICS, solo puede utilizar el grupo de compartición de colas al que está conectado el sistema CICS . Todavía debe especificar el parámetro **QMgrName** , pero su valor se ignora; los caracteres en blanco son una opción adecuada.

Atención: IMS no puede conectarse a un grupo de compartición de colas.

Aplicaciones IBM MQ MQI client: Para aplicaciones IBM MQ MQI client , se intenta una conexión para cada definición de canal de conexión de cliente con el nombre de gestor de colas especificado, hasta que una sea satisfactoria. Sin embargo, el gestor de colas debe tener el mismo nombre que el nombre especificado. Si se especifica un nombre todo en blanco, se intenta cada canal de conexión de cliente con un nombre de gestor de colas todo en blanco hasta que uno sea satisfactorio; en este caso, no hay ninguna comprobación con respecto al nombre real del gestor de colas.

Las aplicaciones cliente de IBM MQ no están soportadas en z/OS, pero z/OS puede actuar como un servidor de IBM MQ , al que se pueden conectar las aplicaciones cliente de IBM MQ .

Grupos de gestores de colas de IBM MQ MQI client: si el nombre especificado empieza con un asterisco (*), el gestor de colas con el que se realiza la conexión puede tener un nombre distinto del especificado por la aplicación. El nombre especificado (sin el asterisco) define un *grupo* de gestores de colas que son elegibles para la conexión. La implementación selecciona uno del grupo intentando cada uno a su vez hasta que se encuentra uno con el que se puede realizar una conexión. El orden en el que se intentan las conexiones está influenciado por el peso del canal de cliente y los valores de afinidad de conexión de los canales candidatos. Si ninguno de los gestores de colas del grupo está disponible para la conexión, la llamada falla. Cada gestor de colas se intenta una sola vez. Si se especifica un asterisco solo para el nombre, se utiliza un grupo de gestores de colas predeterminado definido por la implementación.

Los grupos de gestores de colas sólo están soportados para las aplicaciones que se ejecutan en un entorno de cliente MQ; la llamada falla si una aplicación no cliente especifica un nombre de gestor de colas que empieza con un asterisco. Un grupo se define proporcionando varias definiciones de canal de conexión de cliente con el mismo nombre de gestor de colas (el nombre especificado

sin el asterisco), para comunicarse con cada uno de los gestores de colas del grupo. El grupo predeterminado se define proporcionando una o más definiciones de canal de conexión de cliente, cada una con un nombre de gestor de colas en blanco (por lo tanto, especificar un nombre en blanco tiene el mismo efecto que especificar un solo asterisco para el nombre de una aplicación cliente).

Después de conectarse a un gestor de colas de un grupo, una aplicación puede especificar espacios en blanco de la forma habitual en los campos de nombre de gestor de colas en los descriptores de mensaje y objeto para indicar el nombre del gestor de colas al que se ha conectado la aplicación (el *gestor de colas local*). Si la aplicación necesita conocer este nombre, utilice la llamada MQINQ para consultar el atributo del gestor de colas **QMgrName** .

El prefijo de un asterisco al nombre de conexión implica que la aplicación no depende de la conexión con un gestor de colas determinado del grupo. Las aplicaciones adecuadas son:

- Aplicaciones que colocan mensajes pero no obtienen mensajes.
- Aplicaciones que colocan mensajes de solicitud y, a continuación, obtienen los mensajes de respuesta de una cola *dinámica temporal* .

Las aplicaciones no adecuadas son las que necesitan obtener mensajes de una cola determinada en un gestor de colas determinado; estas aplicaciones no deben añadir un asterisco al nombre.

Si especifica un asterisco, la longitud máxima del resto del nombre es de 47 caracteres.

La longitud de este parámetro la proporciona MQ_Q_MGR_NAME_LENGTH.

Hconn

Tipo: MQHCONN-salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

IBM MQ ahora proporciona la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que cuando se realiza una llamada MQI que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si es una conexión de cliente o servidor y, a continuación, se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a un *Hconn* ahora se puede enlazar con la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de descriptor de contexto: El ámbito del descriptor de contexto devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del descriptor de contexto también depende de la opción MQCNO_HANDLE_SHARE_ * especificada en el campo *Options* de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el descriptor de contexto devuelto es un descriptor de contexto *no compartido* .

  El ámbito de un descriptor de contexto no compartido es la unidad más pequeña de proceso paralelo soportada por la plataforma en la que se ejecuta la aplicación (consulte Tabla 546 en la página 673 para obtener detalles); el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada.
- Si especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el descriptor de contexto devuelto es un descriptor de contexto *compartido* .

  El ámbito de un descriptor de contexto compartido es el proceso que es propietario de la hebra desde la que se ha emitido la llamada; el descriptor de contexto se puede utilizar desde cualquier hebra que pertenezca a dicho proceso. No todas las plataformas soportan hebras.
- Si la llamada MQCONN o MQCONNX falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.

Tabla 546. Ámbito de descriptores de contexto no compartidos en diversas plataformas	
Plataforma	**Ámbito de descriptor de contexto no compartido**
z/OS	• CICS: la tarea CICS • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS por lotes y TSO: la tarea (excluyendo las subtareas de la tarea)
IBM i	Trabajo
UNIX	Hebra
Aplicaciones Windows de 32 bits	Hebra
Aplicaciones Windows de 64 bits	Hebra

En aplicaciones z/OS for CICS , el valor devuelto es:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') La aplicación ya está conectada.

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') No se puede cargar la salida de carga de trabajo de clúster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL ya se ha inicializado.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') No se ha podido cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Módulo de definición de subsistema de adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') No se ha podido cargar el módulo de definición de subsistema de adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTER_STORAGE_INSUFICIENTE
(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Otro gestor de colas ya está conectado.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') El identificador de conexión ya está en uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_ERROR
(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Se produce en una llamada MQCONN o MQCONNX cuando el gestor de colas no puede proporcionar una conexión del tipo de conexión solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Error de configuración de hardware criptográfico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') El coordinador de recuperación existe.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

Además, en la llamada MQCONNX, pasando el bloque de control "MQCSP-Parámetros de seguridad" en la página 336 desde una aplicación CICS o IMS .

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Discrepancia entre la instalación del gestor de colas y la biblioteca seleccionada.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') El repositorio de claves no es válido.

MQRC_MAX_CONNS_LIMIT_ALCANZADO
(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. El gestor de colas con el que se realiza la conexión utilizando la llamada MQCONN se denomina *gestor de colas local*.
2. Las colas que son propiedad del gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas compartidas que son propiedad del grupo de compartición de colas al que pertenece el gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas que son propiedad de gestores de colas remotos aparecen como colas remotas. Es posible colocar mensajes en estas colas, pero no obtener mensajes de estas colas.
3. Si el gestor de colas falla mientras una aplicación se está ejecutando, la aplicación debe volver a emitir la llamada MQCONN para obtener un nuevo descriptor de conexión para utilizarlo en las llamadas IBM MQ posteriores. La aplicación puede emitir la llamada MQCONN periódicamente hasta que la llamada sea satisfactoria.

   Si una aplicación no está segura de si está conectada al gestor de colas, la aplicación puede emitir de forma segura una llamada MQCONN para obtener un descriptor de conexión. Si la aplicación ya está conectada, el descriptor de contexto devuelto es el mismo que el devuelto por la llamada MQCONN anterior, pero con el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED.
4. Cuando la aplicación ha terminado de utilizar las llamadas IBM MQ , la aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.

5. Si la llamada MQCONN falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.
6. En z/OS:
   - Las aplicaciones por lotes, TSO y IMS deben emitir la llamada MQCONN para utilizar las otras llamadas IBM MQ . Estas aplicaciones pueden conectarse a más de un gestor de colas simultáneamente.

     Si el gestor de colas falla, la aplicación debe volver a emitir la llamada después de que el gestor de colas se haya reiniciado para obtener un nuevo descriptor de conexión.

     Aunque las aplicaciones IMS pueden emitir la llamada MQCONN repetidamente, aunque ya estén conectadas, esto no se recomienda para los programas de proceso de mensajes en línea (MPP).
   - Las aplicaciones CICS no tienen que emitir la llamada MQCONN para utilizar las otras llamadas IBM MQ , pero pueden hacerlo si lo desean; se aceptan tanto la llamada MQCONN como la llamada MQDISC. Sin embargo, no es posible conectarse a más de un gestor de colas simultáneamente.

     Si el gestor de colas falla, estas aplicaciones se vuelven a conectar automáticamente cuando se reinicia el gestor de colas, por lo que no es necesario emitir la llamada MQCONN.
7. En z/OS, para definir los gestores de colas disponibles:
   - Para aplicaciones por lotes, los programadores del sistema pueden utilizar la macro CSQBDEF para crear un módulo (CSQBDEFV) que defina el nombre predeterminado del gestor de colas o el nombre del grupo de compartición de colas.
   - Para las aplicaciones IMS , los programadores del sistema pueden utilizar la macro CSQQDEFX para crear un módulo (CSQQDEFV) que defina los nombres de los gestores de colas disponibles y especifique el gestor de colas predeterminado.

     Además, cada gestor de colas debe estar definido en la región de control de IMS y en cada región dependiente que acceda a ese gestor de colas. Para ello, debe crear un miembro de subsistema en IMS.Biblioteca PROCLIB e identifique el miembro del subsistema en las regiones IMS aplicables. Si una aplicación intenta conectarse a un gestor de colas que no está definido en el miembro del subsistema para su región IMS , la aplicación termina de forma anómala.

   z/OS Para obtener más información sobre cómo utilizar estas macros, consulte Macros destinadas al uso del cliente.
8. En IBM i, los programas que finalizan de forma anómala no se desconectan automáticamente del gestor de colas. Escriba aplicaciones para permitir la posibilidad de que la llamada MQCONN o MQCONNX devuelva el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED. Utilice el descriptor de conexión devuelto en esta situación como normal.

Invocación en C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN   Hconn;     /* Connection handle */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Name of queue manager
 01  QMGRNAME  PIC X(48).
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl QMgrName  char(48);       /* Name of queue manager */
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN     DS  F     Connection handle
COMPCODE  DS  F     Completion code
REASON    DS  F     Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim QMgrName As String*48 'Name of queue manager'
Dim Hconn    As Long      'Connection handle'
Dim CompCode As Long      'Completion code'
Dim Reason   As Long      'Reason code qualifying CompCode'
```

MQCONNX - Conectar gestor de colas (ampliado)

La llamada MQCONNX conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas IBM MQ posteriores.

La llamada MQCONNX es como la llamada MQCONN, excepto que MQCONNX permite especificar opciones para controlar la forma en que funciona la llamada.

- Esta llamada está soportada en todos los sistemas IBM MQ y clientes IBM MQ conectados a estos sistemas.

No se puede realizar una conexión de cliente en una instalación sólo de servidor, y no se puede realizar una conexión local en una instalación sólo de cliente.

Sintaxis

MQCONNX (*QMgrName*, *ConnectOpts*, *Hconn*, *CompCode*, *Razón*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Consulte el parámetro **QMgrName** descrito en "MQCONN-Conectar gestor de colas" en la página 670 para obtener más detalles.

ConnectOpts

Tipo: MQCNO-entrada/salida

Consulte "MQCNO - Opciones de conexión" en la página 316 para obtener los detalles.

Hconn

Tipo: MQHCONN-salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

IBM MQ ahora proporciona la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que cuando se realiza una llamada MQI que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si es una conexión de cliente o servidor y, a continuación, se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a un *Hconn* ahora se puede enlazar con la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de descriptor de contexto: El ámbito del descriptor de contexto devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del descriptor de contexto también depende de la opción MQCNO_HANDLE_SHARE_ * especificada en el campo *Options* de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el descriptor de contexto devuelto es un descriptor de contexto *no compartido* .

  El ámbito de un descriptor de contexto no compartido es la unidad más pequeña de proceso paralelo soportada por la plataforma en la que se ejecuta la aplicación (consulte Tabla 547 en la página 678 para obtener detalles); el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada.
- Si especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el descriptor de contexto devuelto es un descriptor de contexto *compartido* .

  El ámbito de un descriptor de contexto compartido es el proceso que es propietario de la hebra desde la que se ha emitido la llamada; el descriptor de contexto se puede utilizar desde cualquier hebra que pertenezca a dicho proceso. No todas las plataformas soportan hebras.
- Si la llamada MQCONN o MQCONNX falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.

Tabla 547. Ámbito de descriptores de contexto no compartidos en diversas plataformas

Plataforma	Ámbito de descriptor de contexto no compartido
z/OS	• CICS: la tarea CICS • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS por lotes y TSO: la tarea (excluyendo las subtareas de la tarea)

Tabla 547. Ámbito de descriptores de contexto no compartidos en diversas plataformas (continuación)	
Plataforma	**Ámbito de descriptor de contexto no compartido**
IBM i	Trabajo
UNIX	Hebra
Aplicaciones Windows de 32 bits	Hebra
Aplicaciones Windows de 64 bits	Hebra

En aplicaciones z/OS for CICS , el valor devuelto es:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

CompCode
Tipo: MQLONG - salida

Consulte el parámetro **CompCode** descrito en "MQCONN-Conectar gestor de colas" en la página 670 para obtener más detalles.

Razón
Tipo: MQLONG - salida

Las llamadas MQCONN y MQCONNX pueden devolver los códigos siguientes. Para obtener una lista de códigos adicionales que puede devolver la llamada MQCONNX, consulte los códigos siguientes.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') La aplicación ya está conectada.

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') No se puede cargar la salida de carga de trabajo de clúster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL ya se ha inicializado.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') No se ha podido cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Módulo de definición de subsistema de adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') No se ha podido cargar el módulo de definición de subsistema de adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTER_STORAGE_INSUFICIENTE
(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Otro gestor de colas ya está conectado.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') El identificador de conexión ya está en uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_ERROR
(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Se produce en una llamada MQCONN o MQCONNX cuando el gestor de colas no puede proporcionar una conexión del tipo de conexión solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Error de configuración de hardware criptográfico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') El coordinador de recuperación existe.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

Además, en la llamada MQCONNX, pasando el bloque de control “MQCSP-Parámetros de seguridad” en la página 336 desde una aplicación CICS o IMS .

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Discrepancia entre la instalación del gestor de colas y la biblioteca seleccionada.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') El repositorio de claves no es válido.

MQRC_MAX_CONNS_LIMIT_ALCANZADO
(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

La llamada MQCONNX puede devolver los siguientes códigos de razón adicionales:

Si *CompCode* es MQCC_FAILED:

MQRC_AIR_ERROR
(2385, X' 951 ') El registro de información de autenticación no es válido.

MQRC_AUTH_INFO_CONN_NAME_ERROR
(2387, X' 953 ') Nombre de conexión de información de autenticación no válido.

MQRC_AUTH_INFO_REC_COUNT_ERROR
(2383, X'94F') El recuento de registros de información de autenticación no es válido.

MQRC_AUTH_INFO_REC_ERROR
(2384, X' 950 ') Los campos de registro de información de autenticación no son válidos.

MQRC_AUTH_INFO_TYPE_ERROR
(2386, X' 952 ') Tipo de información de autenticación no válido.

MQRC_CD_ERROR
(2277, X'8E5') Definición de canal no válida.

MQRC_CLIENT_CONN_ERROR
(2278, X'8E6') Los campos de conexión de cliente no son válidos.

MQRC_CNO_ERROR
(2139, X'85B') La estructura de opciones de conexión no es válida.

MQRC_CONN_TAG_IN_USE
(2271, X'8DF') Código de conexión en uso.

MQRC_CONN_TAG_NOT_USABLE
(2350, X'92E') Código de conexión no utilizable.

MQRC_LDAP_PASSWORD_ERROR
(2390, X' 956 ') La contraseña LDAP no es válida.

MQRC_LDAP_USER_NAME_ERROR
(2388, X' 954 ') Los campos de nombre de usuario LDAP no son válidos.

MQRC_LDAP_USER_NAME_LENGTH_ERR
(2389, X' 955 ') La longitud del nombre de usuario LDAP no es válida.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_SCO_ERROR
(2380, X'94C') La estructura de opciones de configuración SSL no es válida.

MQRC_SSL_CONFIG_ERROR
(2392, X' 958 ') Error de configuración SSL.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

Para el lenguaje de programación Visual Basic, se aplica el punto siguiente:

- El parámetro **ConnectOpts** se declara como de tipo MQCNO. Si la aplicación se ejecuta como un IBM MQ MQI clienty desea especificar los parámetros del canal de conexión de cliente, declare el parámetro **ConnectOpts** como de tipo Any, para que la aplicación pueda especificar una estructura MQCNOCD en la llamada en lugar de una estructura MQCNO. Sin embargo, esto significa que no se puede comprobar el parámetro **ConnectOpts** para asegurarse de que es el tipo de datos correcto.

Invocación en C

```
MQCONNX (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;     /* Name of queue manager */
MQCNO     ConnectOpts;  /* Options that control the action of MQCONNX */
MQHCONN   Hconn;        /* Connection handle */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCONNX' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Name of queue manager
 01  QMGRNAME     PIC X(48).
**   Options that control the action of MQCONNX
 01  CONNECTOPTS.
     COPY CMQCNOV.
**   Connection handle
 01  HCONN        PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE     PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON       PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCONNX (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl QMgrName     char(48);       /* Name of queue manager */
dcl ConnectOpts  like MQCNO;     /* Options that control the action of
                                    MQCONNX */
dcl Hconn        fixed bin(31);  /* Connection handle */
```

```
dcl CompCode     fixed bin(31);  /* Completion code */
dcl Reason       fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCONNX,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
QMGRNAME     DS       CL48  Name of queue manager
CONNECTOPTS  CMQCNOA  ,     Options that control the action of MQCONNX
HCONN        DS       F     Connection handle
COMPCODE     DS       F     Completion code
REASON       DS       F     Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCONNX QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim QMgrName    As String*48 'Name of queue manager'
Dim ConnectOpts As MQCNO     'Options that control the action of'
                             'MQCONNX'
Dim Hconn       As Long      'Connection handle'
Dim CompCode    As Long      'Completion code'
Dim Reason      As Long      'Reason code qualifying CompCode'
```

MQCRTMH-Crear manejador de mensajes

La llamada MQCRTMH devuelve un descriptor de mensaje.

Una aplicación puede utilizar la llamada MQCRTMH en llamadas de cola de mensajes posteriores:

- Utilice la llamada MQSETMP para establecer una propiedad del manejador de mensajes.
- Utilice la llamada MQINQMP para consultar el valor de una propiedad del manejador de mensajes.
- Utilice la llamada MQDLTMP para suprimir una propiedad del manejador de mensajes.

El descriptor de mensaje se puede utilizar en las llamadas MQPUT y MQPUT1 para asociar las propiedades del descriptor de mensaje con las del mensaje que se está colocando. De forma similar, al especificar un descriptor de mensaje en la llamada MQGET, se puede acceder a las propiedades del mensaje que se está recuperando utilizando el descriptor de mensaje cuando se completa la llamada MQGET.

Utilice MQDLTMH para suprimir el descriptor de mensaje.

Sintaxis

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior. Si la conexión con el gestor de colas deja de ser válida y no hay ninguna llamada IBM MQ operativa en el manejador de mensajes, se llama implícitamente a MQDLTMH para suprimir el mensaje.

De forma alternativa, puede especificar el valor siguiente:

MQHC_UNASSOCIATED_HCONN
El descriptor de conexión no representa una conexión con ningún gestor de colas en particular.

Cuando se utiliza este valor, el descriptor de mensaje debe suprimirse con una llamada explícita a MQDLTMH para liberar cualquier almacenamiento asignado a él; IBM MQ nunca suprime implícitamente el descriptor de mensaje.

Debe haber al menos una conexión válida con un gestor de colas establecido en la hebra que crea el manejador de mensajes; de lo contrario, la llamada falla con MQRC_HCONN_ERROR.

En un entorno con varias instalaciones en un único sistema, el valor MQHC_UNASSOCIATED_HCONN está limitado a utilizar con la primera instalación cargada en el proceso. Se devuelve el código de razón MQRC_HMSG_NOT_AVAILABLE si el manejador de mensajes se proporciona a una instalación diferente.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_CONN
Descriptor de conexión predeterminado

CrtMsgHOpts
Tipo: MQCMHO-entrada

Las opciones que controlan la acción de MQCRTMH. Consulte MQCMHO para obtener detalles.

Hmsg
Tipo: MQHMSG-salida

En la salida, se devuelve un descriptor de mensaje que se puede utilizar para establecer, consultar y suprimir propiedades del descriptor de mensaje. Inicialmente, el manejador de mensajes no contiene propiedades.

Un descriptor de mensaje también tiene un descriptor de mensaje asociado. Inicialmente contiene los valores predeterminados. Los valores de los campos de descriptor de mensaje asociados se pueden establecer y consultar utilizando las llamadas MQSETMP y MQINQMP. La llamada MQDLTMP restablece un campo del descriptor de mensaje a su valor predeterminado.

Si el parámetro *Hconn* se especifica como el valor MQHC_UNASSOCIATED_HCONN, el descriptor de mensaje devuelto se puede utilizar en llamadas MQGET, MQPUT o MQPUT1 con cualquier conexión dentro de la unidad de proceso, pero sólo puede estar en uso por una llamada IBM MQ cada vez. Si el descriptor de contexto está en uso cuando una segunda llamada IBM MQ intenta utilizar el mismo descriptor de contexto de mensaje, la segunda llamada IBM MQ falla con el código de razón MQRC_MSG_HANDLE_IN_USE.

Si el parámetro *Hconn* no es MQHC_UNASSOCIATED_HCONN, el manejador de mensajes devuelto sólo se puede utilizar en la conexión especificada.

Se debe utilizar el mismo valor de parámetro *Hconn* en las llamadas MQI posteriores donde se utiliza este manejador de mensajes:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

El descriptor de mensaje devuelto deja de ser válido cuando se emite la llamada MQDLTMH para el descriptor de mensaje, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. MQDLTMH se llama implícitamente si se proporciona una conexión específica cuando se

crea el descriptor de mensaje y la conexión con el gestor de colas deja de ser válida, por ejemplo, si se llama a MQDBC.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CMHO_ERROR
(2461, X'099D') Crear estructura de opciones de manejador de mensajes no válida.

MQRC_CONNECTION_BROKEN
(2273, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'07E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;       /* Connection handle */
MQCMHO   CrtMsgHOpts; /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;        /* Message handle */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

COBOL

```
CALL 'MQCRTMH' USING HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Options that control the action of MQCRTMH
 01  CRTMSGHOPTS.
     COPY CMQCMHOV.
**   Message handle
 01  HMSG    PIC S9(18) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

PL/I

```
call MQCRTMH (Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl CrtMsgHOpts like MQCMHO;    /* Options that control the action of MQCRTMH */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

High Level Assembler

```
CALL MQCRTMH,(HCONN,CRTMSGHOPTS,HMSG,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN          DS        F  Connection handle
CRTMSGHOPTS    CMQCMHOA  ,  Options that control the action of MQCRTMH
HMSG           DS        D  Message handle
COMPCODE       DS        F  Completion code
REASON         DS        F  Reason code qualifying COMPCODE
```

MQCTL-devoluciones de llamada de control

La llamada MQCTL realiza acciones de control en devoluciones de llamada y los manejadores de objeto abiertos para una conexión.

Sintaxis

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y puede especificar el siguiente valor especial para *Hconn* :

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Operación

Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una, y sólo una, de las opciones siguientes:

MQOP_START

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Las devoluciones de llamada se ejecutan en una hebra iniciada por el sistema, que es diferente de cualquiera de las hebras de aplicación.

Esta operación proporciona el control del descriptor de conexión proporcionado al sistema. Las únicas llamadas MQI que puede emitir una hebra que no sea la hebra de consumidor son:

- MQCTL con operación MQOP_STOP
- MQCTL con la operación MQOP_SUSPEND
- MQDISC-Realiza MQCTL con la operación MQOP_STOP antes de desconectar el HConn.

Se devuelve MQRC_HCONN_ASYNC_ACTIVE si se emite una llamada de API de IBM MQ mientras se inicia el descriptor de conexión y la llamada no se origina en una función de consumidor de mensajes.

Si un consumidor de mensajes detiene la conexión durante MQCBCT_START_CALL, la llamada MQCTL devuelve un código de razón de anomalía de MQRC_CONNECTION_STOPPED.

Esto se puede emitir en una función de consumidor. Para la misma conexión que la rutina de devolución de llamada, su única finalidad es cancelar una operación MQOP_STOP emitida anteriormente.

Esta opción no está soportada en los entornos siguientes: CICS en z/OS o si la aplicación está enlazada con una biblioteca IBM MQ sin hebras.

MQOP_START_WAIT

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Los consumidores de mensajes se ejecutan en la misma hebra y el control no se devuelve al interlocutor de MQCTL hasta que:

- Liberado por el uso de las operaciones MQCTL MQOP_STOP o MQOP_SUSPEND, o
- Todas las rutinas de consumidor se han desregistrado o suspendido.

Si se anula el registro o se suspenden todos los consumidores, se emite una operación MQOP_STOP implícita.

Esta opción no se puede utilizar desde una rutina de devolución de llamada, ya sea para el descriptor de conexión actual o cualquier otro descriptor de conexión. Si se intenta la llamada, se devuelve con MQRC_ENVIRONMENT_ERROR.

Si, en algún momento durante una operación MQOP_START_WAIT no hay ningún consumidor registrado, no suspendido, la llamada falla con un código de razón de MQRC_NO_CALLBACKS_ACTIVE.

Si, durante una operación MQOP_START_WAIT, la conexión se suspende, la llamada MQCTL devuelve un código de razón de aviso de MQRC_CONNECTION_SUSPENDED; la conexión permanece 'iniciada'.

La aplicación puede elegir emitir MQOP_STOP o MQOP_RESUME. En esta instancia, la operación MQOP_RESUME se bloquea.

Esta opción no está soportada en un cliente de una sola hebra.

MQOP_STOP
Detenga el consumo de mensajes y espere a que todos los consumidores completen sus operaciones antes de que se complete esta opción. Esta operación libera el descriptor de conexión.

Si se emite desde dentro de una rutina de devolución de llamada, esta opción no entra en vigor hasta que se cierra la rutina. No se llama a más rutinas de consumidor de mensajes después de que se hayan completado las rutinas de consumidor para los mensajes que ya se han leído, y después de que se hayan realizado llamadas de detención (si se han solicitado) a rutinas de devolución de llamada.

Si se emite fuera de una rutina de devolución de llamada, el control no vuelve al llamante hasta que se hayan completado las rutinas del consumidor para los mensajes ya leídos y después de que se hayan realizado las llamadas de detención (si se han solicitado) a las devoluciones de llamada. Sin embargo, las devoluciones de llamada permanecen registradas.

Esta función no tiene ningún efecto en los mensajes de lectura anticipada. Debe asegurarse de que los consumidores ejecuten MQCLOSE (MQCO_QUIESCE), desde dentro de la función de devolución de llamada, para determinar si hay más mensajes disponibles para entregar.

MQOP_SUSPENDER
Pausar el consumo de mensajes. Esta operación libera el descriptor de conexión.

Esto no tiene ningún efecto en la lectura anticipada de los mensajes para la aplicación. Si tiene previsto dejar de consumir mensajes durante mucho tiempo, considere la posibilidad de cerrar la cola y volver a abrirla cuando continúe el consumo.

Si se emite desde dentro de una rutina de devolución de llamada, no entra en vigor hasta que se cierra la rutina. No se llamarán más rutinas de consumidor de mensajes después de las salidas de rutina actuales.

Si se emite fuera de una devolución de llamada, el control no vuelve al interlocutor hasta que se haya completado la rutina de consumidor actual y no se llame a más.

MQOP_RESUME
Reanude el consumo de mensajes.

Esta opción se emite normalmente desde la hebra de aplicación principal, pero también se puede utilizar desde una rutina de devolución de llamada para cancelar una solicitud de suspensión anterior emitida en la misma rutina.

Si se utiliza MQOP_RESUME para reanudar un MQOP_START_WAIT, la operación se bloquea.

ControlOpts
Tipo: MQCTLO-entrada

Opciones que controlan la acción de MQCTL

Consulte MQCTLO para obtener detalles de la estructura.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') No se puede llamar a la rutina de devolución de llamada

MQRC_CALLBACK_NOT_ REGISTRADO
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Se ha especificado CallbackFunction y CallbackName en una llamada MQOP_REGISTER.

O bien se ha especificado CallbackFunction o CallbackName pero no coincide con la función de devolución de llamada registrada actualmente.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Campo de tipo CallBackincorrecto.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CBD_ERROR
(2444, X'98C') El bloque de opciones es incorrecto.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B5') Campo de longitud MaxMsgincorrecto

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') No se ha podido encontrar el punto de entrada de función especificado en el módulo.

MQRC_MODULE_INVALID
(2496, X'9C0') Se ha encontrado el módulo pero es del tipo incorrecto (32 bit/64 bit) o no es una dll válida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

MQRC_MSG_ID_ERROR
(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPERATION_ERROR
(2488, X'9B8') Código de operación incorrecto en llamada de API

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Las rutinas de devolución de llamada deben comprobar las respuestas de todos los servicios que invocan y, si la rutina detecta una condición que no se puede resolver, debe emitir un mandato MQCB MQOP_DEREGISTER para evitar llamadas repetidas a la rutina de devolución de llamada.
2. Si está utilizando el consumo asíncrono en una aplicación en la que un gestor de transacciones XA está gestionando transacciones globales, incluidas las actualizaciones de IBM MQ, debe tener en cuenta los siguientes puntos adicionales:
   a. No es válido llamar a MQCTL (MQOP_START) para un **HConn**, después de que se haya creado, después de llamar a **xa_open**.

      La razón es que **HConn** se ha conectado a un contexto XA y, por lo tanto, no se puede acceder a él en la hebra o hebras separadas, en uso por el mecanismo de consumo asíncrono.
   b. Si llama a MQCTL (MQOP_START) en ese escenario, la llamada falla con el código de razón MQRC_ASYNC_XA_CONFLICT (2350).
   c. Es válido llamar a MQCTL (MQOP_START_WAIT) para un **HConn**, después de que se haya creado, después de llamar a **xa_open**.

      La razón es que este método de inicio del mecanismo de consumo asíncrono hace que todas las devoluciones de llamada adicionales para **HConn** se ejecuten en la hebra donde se realiza la llamada MQCTL. Por lo tanto, el enlace entre **HConn** y la hebra no se pierde.
3. z/OS En z/OS, cuando la operación es MQOP_START:
   - Los programas que utilizan rutinas de devolución de llamada asíncrona deben estar autorizados para utilizar z/OS UNIX System Services (USS).
   - Los programas LE (Language Environment) que utilizan rutinas de devolución de llamada asíncrona deben utilizar la opción de tiempo de ejecución LE POSIX(ON).
   - Los programas no LE que utilizan rutinas de devolución de llamada asíncronas no deben utilizar la interfaz pthread_create de USS (servicio invocable BPX1PTC).
4. z/OS MQCTL no está soportado en el adaptador IMS .

Nota: En CICS, MQOP_START no está soportado. En su lugar, utilice la llamada de función MQOP_START_WAIT.

Invocación en C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQLONG   Operation;     /* Operation being processed */
MQCTLO   ControlOpts    /* Options that control the action of MQCTL */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Operation
 01  OPERATION PIC S9(9) BINARY.
**   Control Options
 01  CTLOPTS.
     COPY CMQCTLOV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Operation    fixed bin(31); /* Operation */
dcl CtlOpts like MQCTLO;        /* Options that control the action of MQCTL */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

MQDISC-Desconectar gestor de colas

La llamada MQDISC interrumpe la conexión entre el gestor de colas y el programa de aplicación, y es la inversa de la llamada MQCONN o MQCONNX.

- En z/OS, todas las aplicaciones que utilizan el consumo de mensajes asíncronos, el manejo de sucesos o la devolución de llamada, la hebra de control principal debe emitir una llamada MQDISC antes de finalizar. Consulte Consumo asíncrono de mensajes de IBM MQ para obtener más detalles.
- En z/OS, las aplicaciones CICS no necesitan emitir esta llamada para desconectarse del gestor de colas.

  Si una aplicación CICS realiza esta llamada, no tendrá ningún efecto a menos que se haya realizado una llamada MQCONNX anterior, especificando uno de los siguientes:

  MQCNO_SERIALIZE_CONN_TAG_Q_MGR
  MQCNO_SERIALIZE_CONN_TAG_QSG
  MQCNO_RESTRICT_CONN_TAG_Q_MGR o
  MQCNO_RESTRICT_CONN_TAG_QSG

  , en cuyo caso se cierran todos los manejadores de objetos abiertos actualmente.

Sintaxis

MQDISC (*Hconn*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN-entrada/salida

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS puede omitir la llamada MQCONN y especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Al finalizar correctamente la llamada, el gestor de colas establece *Hconn* en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

MQHC_UNUSABLE_HCONN
Descriptor de conexión inutilizable.

En z/OS, *Hconn* se establece en un valor que no está definido.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los códigos siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_CONN_TAG_NOT_LIBERADO
(2344, X' 928 ') No se ha liberado la etiqueta de conexión.

MQRC_OUTCOME_PENDING
(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_DISC_LOAD_ERROR
(2138, X'85A') No se ha podido cargar el módulo de desconexión del adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Si se emite una llamada MQDISC cuando la conexión todavía tiene objetos abiertos bajo esa conexión, el gestor de colas cierra esos objetos, con las opciones de cierre establecidas en MQCO_NONE.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de cómo finalice la aplicación:
   a. Si la aplicación emite la llamada MQDISC antes de finalizar:
      - Para una unidad de trabajo coordinada por el gestor de colas, el gestor de colas emite la llamada MQCMIT en nombre de la aplicación. La unidad de trabajo se confirma si es posible y se restituye si no es así.
      - Para una unidad de trabajo coordinada externamente, no hay ningún cambio en el estado de la unidad de trabajo; sin embargo, el gestor de colas normalmente indica que la unidad de trabajo debe confirmarse cuando se lo solicite el coordinador de la unidad de trabajo.

        En z/OS, CICS, IMS (distintos de los programas DL/1 por lotes) y las aplicaciones RRS son así.

   b. Si la aplicación finaliza normalmente pero sin emitir la llamada MQDISC, la acción realizada depende del entorno:
      - En z/OS, excepto para las aplicaciones MQ Java o MQ JMS , se producen las acciones descritas en la nota 2a .
      - En todos los demás casos, se producen las acciones descritas en la nota 2c .

      Debido a las diferencias entre entornos, asegúrese de que las aplicaciones que desea portar confirmen o restituya la unidad de trabajo antes de que finalicen.

   c. Si la aplicación finaliza *anormalmente* sin emitir la llamada MQDISC, la unidad de trabajo se restituye.
3. En z/OS, se aplican los puntos siguientes:
   - Las aplicaciones CICS no tienen que emitir la llamada MQDISC para desconectarse del gestor de colas, porque el propio sistema CICS se conecta al gestor de colas y la llamada MQDISC no tiene ningún efecto en esta conexión.

- Las aplicaciones CICS, IMS (que no sean programas DL/1 por lotes) y RRS utilizan unidades de trabajo coordinadas por un coordinador de unidad de trabajo externo. Como resultado, la llamada MQDISC no afecta al estado de la unidad de trabajo (si existe) que existe cuando se emite la llamada.

  Sin embargo, la llamada MQDISC *sí* indica el final de uso del código de conexión *ConnTag* que se ha asociado con la conexión mediante una llamada MQCONNX anterior emitida por la aplicación. Si hay una unidad de trabajo activa que hace referencia al código de conexión cuando se emite la llamada MQDISC, la llamada se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_CONN_TAG_NOT_LIBERAR. El código de conexión no pasa a estar disponible para su reutilización hasta que el coordinador de unidad de trabajo externo haya resuelto la unidad de trabajo.

Nota: En CICS, MQOP_START no está soportado. En su lugar, utilice la llamada de función MQOP_START_WAIT.

Invocación en C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación de ensamblador System/390

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQDLTMH-Suprimir descriptor de mensaje

La llamada MQDLTMH suprime un manejador de mensajes y es el inverso de la llamada MQCRTMH.

Sintaxis

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el manejador de mensajes se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el manejador de mensajes; de lo contrario, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG-entrada/salida

Este es el descriptor de contexto de mensaje que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

Al finalizar correctamente la llamada, el descriptor de contexto se establece en un valor no válido para el entorno. Este valor es:

MQHM_UNUSABLE_HMSG
Manejador de mensajes inutilizable.

El descriptor de mensaje no se puede suprimir si hay otra llamada IBM MQ en curso a la que se ha pasado el mismo descriptor de mensaje.

DltMsgHOpts

Tipo: MQDMHO-entrada

Consulte MQDMHO para obtener más detalles.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMHO_ERROR
(2462, X'099E') La estructura de opciones del manejador de mensajes no es válida.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Invocación en C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;       /* Connection handle */
MQHMSG   Hmsg;        /* Message handle */
MQDMHO   DltMsgHOpts; /* Options that control the action of MQDLTMH */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGHOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**  Connection handle
01   HCONN   PIC S9(9) BINARY.

**  Options that control the action of MQDLTMH
01   DLTMSGHOPTS.
COPY CMQDMHOL.

**  Completion code
01  COMPCODE   PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01   REASON      PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          /* 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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMH,(HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN          DS        F  Connection handle
HMSG           DS        D  Message handle
DLTMSGHOPTS    CMQDMHOA  ,  Options that control the action of MQDLTMH
COMPCODE       DS        F  Completion code
REASON         DS        F  Reason code qualifying COMPCODE
```

MQDLTMP-Suprimir propiedad de mensaje

La llamada MQDLTMP suprime una propiedad de un manejador de mensajes y es la inversa de la llamada MQSETMP.

Sintaxis

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Nombre*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje, de lo contrario, la llamada fallará con MQRC_CONNECTION_BROKEN.

Hmsg
Tipo: MQHMSG-entrada

Este es el descriptor de contexto de mensaje que contiene la propiedad que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

DltPropOpts
Tipo: MQDMPO-entrada

Consulte el tipo de datos MQDMPO para obtener detalles.

Nombre
Tipo: MQCHARV-entrada

El nombre de la propiedad que se va a suprimir. Consulte Nombres de propiedad para obtener más información sobre los nombres de propiedad.

Los comodines no están permitidos en el nombre de propiedad.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Propiedad no disponible.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'086D') Los ASID primario y de inicio difieren.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMPO_ERROR
(2481, X'09B1') Suprimir estructura de opciones de propiedad de mensaje no válida.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte:

- Mensajes y códigos de razón para IBM MQ for z/OS
- Códigos de razón y finalización de API para otras plataformas IBM MQ

.

Invocación en C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;       /* Connection handle */
MQHMSG  Hmsg;        /* Message handle */
MQDMPO  DltPropOpts; /* Options that control the action of MQDLTMP */
MQCHARV Name;        /* Property name */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Message handle
 01  HMSG         PIC S9(18) BINARY.
**   Options that control the action of MQDLTMP
 01  DLTPROPOPTS.
     COPY CMQDMPOV.
**   Property name
 01  NAME.
     COPY CMQCHRVV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31); /* Connection handle */
dcl Hmsg        fixed bin(63); /* Message handle */
dcl DltPropOpts like MQDMPO;   /* Options that control the action of MQDLTMP */
dcl Name        like MQCHARV;  /* Property name */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason      fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMP,(HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
DLTPROPOPTS  CMQDMPOA   ,     Options that control the action of MQDLTMP
NAME         CMQCHRVA   ,     Property name
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

Mensaje MQGET - get

La llamada MQGET recupera un mensaje de una cola local que se ha abierto utilizando la llamada MQOPEN.

Sintaxis

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este manejador representa la cola de la que debe recuperarse el mensaje. El valor de *Hobj* ha sido devuelto por una llamada MQOPEN anterior. La cola debe haberse abierto con una o varias de las opciones siguientes (consulte “MQOPEN-Abrir objeto” en la página 742 para obtener más detalles):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado. Consulte “MQMD - Descriptor de mensaje” en la página 421 para obtener los detalles.

Si *BufferLength* es menor que la longitud del mensaje, el gestor de colas rellena *MsgDesc* , si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro **GetMsgOpts** (consulte MQGMO-Campo Options).

Si la aplicación proporciona un MQMD version-1 , el mensaje devuelto tiene un MQMDE con el prefijo de los datos del mensaje de aplicación, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. Si todos los campos de MQMDE tienen valores predeterminados, se omitirá MQMDE. Un nombre de formato de MQFMT_MD_EXTENSION en el campo *Formato* en MQMD indica que hay un MQMDE.

La aplicación no tiene que proporcionar una estructura MQMD si se suministra un manejador de mensajes válido en el campo *MsgHandle*. Si no se proporciona ningún valor en este campo, el descriptor del mensaje se tomará del descriptor asociado a los manejadores de mensajes.

Si la aplicación proporciona un manejador de mensajes en lugar de una estructura MQMD, y especifica MQGMO_PROPERTIES_FORCE_MQRFH2, la llamada fallará con el código de razón MQRC_MD_ERROR. La llamada también falla, con el código de razón MQRC_MD_ERROR, si la aplicación no proporciona una estructura MQMD y especifica MQGMO_PROPERTIES_AS_Q_DEF, y el atributo de cola **PropertyControl** es MQPROP_FORCE_MQRFH2.

Si se especifican opciones de coincidencia y se utiliza el descriptor de mensajes asociado con el manejador de mensajes, los campos de entrada que se utilizan para la comparación proceden del manejador de mensajes.

GetMsgOpts
Tipo: MQGMO - entrada/salida

Consulte “MQGMO-Opciones de obtención de mensajes” en la página 367 para obtener los detalles.

BufferLength
Tipo: MQLONG - entrada

Es la longitud en bytes del área *Buffer* . Especifique cero para los mensajes que no tienen datos o si se va a eliminar el mensaje de la cola y a descartar los datos (en este caso debe especificar MQGMO_ACCEPT_TRUNCATED_MSG).

Nota: La longitud del mensaje más largo que es posible leer de la cola la proporciona el atributo de cola **MaxMsgLength** ; consulte “Atributos para colas” en la página 848.

Almacenamiento intermedio
Tipo: MQBYTExBufferLength - salida

Se trata del área que contiene los datos del mensaje. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un ejemplo que contiene un entero binario de 64 bits podría requerir una alineación de 8 bytes.

Si *BufferLength* es menor que la longitud del mensaje, la mayor parte posible del mensaje se mueve a **Buffer**. Esto sucede si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro **GetMsgOpts** (consulte Campo MQGMO-Options para obtener más información).

El juego de caracteres y la codificación de los datos en **Buffer** se proporcionan mediante los campos *CodedCharSetId* y *Encoding* devueltos en el parámetro **MsgDesc** . Si estos valores son distintos de los valores que necesita el destinatario, éste deberá convertir los datos de mensajes de la aplicación en el juego de caracteres y en la codificación necesarios. Se puede utilizar la opción MQGMO_CONVERT (con una salida escrita por el usuario, si es necesario) para convertir los datos de mensaje; consulte “MQGMO-Opciones de obtención de mensajes” en la página 367 para obtener detalles de esta opción.

Nota: Todos los demás parámetros de la llamada MQGET están en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE).

Si falla la llamada, el contenido del almacenamiento también se podría haber modificado.

En el lenguaje de programación C, el parámetro se declara como pointer-to-void y se puede especificar cualquier tipo de datos como parámetro.

Si el parámetro **BufferLength** es cero, no se hace referencia a *Buffer* ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DataLength
Tipo: MQLONG - salida

Indica la longitud en bytes de los datos de la aplicación *en el mensaje*. Si este valor es mayor que *BufferLength*, sólo se devuelven *BufferLength* bytes en el parámetro **Buffer** (es decir, el mensaje se trunca). Si el valor es cero, el mensaje no contendrá datos de aplicación.

Si *BufferLength* es menor que la longitud del mensaje, el gestor de colas sigue completando *DataLength* , si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro **GetMsgOpts** (consulte MQGMO-Campo Options para obtener más información). Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para introducir los datos del mensaje y después reemitir la llamada con un almacenamiento intermedio que tenga el tamaño adecuado.

Sin embargo, si se especifica la opción MQGMO_CONVERT y los datos de mensaje convertidos son demasiado largos para caber en *Buffer*, el valor devuelto para *DataLength* es:

- La longitud de los datos *sin convertir* , para los formatos definidos por el gestor de colas.

  En este caso, si la naturaleza de los datos hace que se expanda durante la conversión, la aplicación debe asignar un almacenamiento intermedio mayor que el valor devuelto por el gestor de colas para *DataLength*.
- El valor devuelto por la salida de conversión de datos, para formatos definidos por la aplicación.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón listados son los que el gestor de colas puede devolver para el parámetro **Reason** . Si la aplicación especifica la opción MQGMO_CONVERT y se invoca una salida escrita por el usuario para convertir algunos o todos los datos de mensaje, la salida decide qué valor se devuelve para el parámetro **Reason** . Como resultado, es posible tener valores distintos de los que se documentan.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_CONVERTED_STRING_TOO_BIG
(2190, X'88E') La serie convertida es demasiado grande para el campo.

MQRC_DBCS_ERROR
(2150, X'866') La serie DBCS no es válida.

MQRC_FORMAT_ERROR
(2110, X'83E') El formato de mensaje no válido.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_CCSIDS
(2243, X'8C3') Los segmentos del mensaje tienen distintos CCSID.

MQRC_INCONSISTENT_ENCODINGS
(2244, X'8C4') Los segmentos del mensaje tienen distintas codificaciones.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Uso no válido del token de mensaje.

MQRC_NO_MSG_LOCKED
(2209, X'8A1') No hay ningún mensaje bloqueado.

MQRC_NOT_CONVERTED
(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_OPTIONS_CHANGED
(nnnn, X'xxx') Se han cambiado las opciones que tenían que ser coherentes.

MQRC_PARTIALLY_CONVERTED
(2272, X'8E0') Los datos del mensaje se han convertido parcialmente.

MQRC_SIGNAL_REQUEST_ACCEPTED
(2070, X'816') No se ha devuelto ningún mensaje (pero se acepta la solicitud de señal).

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

MQRC_TRUNCATED_MSG_FAILED
(2080, X'820') Se ha devuelto un mensaje truncado (el proceso no se ha completado).

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MSG_ID_ERROR
(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECOND_MARK_NOT_ALLOWED
(2062, X'80E') Ya hay un mensaje marcado.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_SIGNAL1_ERROR
(2099, X'833') Campo de señal no válido.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. El mensaje recuperado se suprime normalmente de la cola. Esta supresión puede producirse como parte de la propia llamada MQGET o como parte de un punto de sincronización.

   Las opciones de examen son: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT y MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Si se especifica la opción MQGMO_LOCK con una de las opciones de examen, el mensaje examinado se bloquea para que sea visible sólo para este manejador.

   Si se especifica la opción MQGMO_UNLOCK, se desbloquea un mensaje anteriormente bloqueado. No se recupera ningún mensaje en este caso y los parámetros **MsgDesc**, **BufferLength**, **Buffer** y **DataLength** no se comprueban ni modifican.

3. Para las aplicaciones que emiten una llamada MQGET, se puede perder el mensaje recuperado si la aplicación termina de norma anormal o si se perdiera la conexión mientras se procesa la llamada. Este problema surge porque el sustituto que se ejecuta en la misma plataforma que el gestor de colas que emite la llamada MQGET en nombre de la aplicación no puede detectar la pérdida de la aplicación hasta que el sustituto está a punto de devolver el mensaje a la aplicación, después de que el mensaje se haya eliminado de la cola. Este problema puede ocurrir tanto para mensajes persistentes como para mensajes no persistentes.

   Para eliminar el riesgo de perder mensajes de esta forma, recupere siempre los mensajes en unidades de trabajo. Es decir, especificando la opción MQGMO_SYNCPOINT en la llamada MQGET y utilizando las llamadas MQCMIT o MQBACK para confirmar o para restituir la unidad de trabajo cuando finaliza el proceso del mensaje. Si se especifica MQGMO_SYNCPOINT y el cliente finaliza de forma anómala o la conexión se pierde, el sustituto restituye la unidad de trabajo en el gestor de colas y el mensaje se restablece en la cola. Para obtener más información sobre los puntos de sincronización, consulte Consideraciones sobre los puntos de sincronización en aplicaciones IBM MQ.

   Esta situación puede producirse con clientes IBM MQ , así como con aplicaciones que se ejecutan en la misma plataforma que el gestor de colas.

4. Si una aplicación pone una secuencia de mensajes en unadeterminada cola dentro de una única unidad de trabajo y, a continuación, confirma esa unidad de trabajo de forma satisfactoria, los mensajes están disponibles para la recuperación tal como se indica a continuación:

   - Si la cola es una *cola no compartida* (es decir, una cola local), todos los mensajes de la unidad de trabajo pasan a estar disponibles al mismo tiempo.
   - Si la cola es una *cola compartida*, los mensajes de la unidad de trabajo pasan a estar disponibles en el orden en el que se colocaron, pero no todos al mismo tiempo. Cuando el sistema está muy cargado, es posible que el primer mensaje de la unidad de trabajo se recupere correctamente, pero que la llamada MQGET para el segundo o posteriores mensajes en la unidad de trabajo fallen con MQRC_NO_MSG_AVAILABLE. Si esto ocurre, la aplicación debe esperar un poco y luego volverá a intentar la operación.

5. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Consulte Notas de uso de MQPUT para obtener detalles. Si las condiciones se cumplen, los mensajes se presentan a la aplicación receptora en el orden en que fueron enviados, si:

   - Sólo un receptor está obteniendo mensajes de la cola.

     Si hay dos o más aplicaciones que obtienen mensajes de la cola, debe ponerse de acuerdo con el remitente sobre el mecanismo que se utilizará para identificar los mensajes que pertenecen a una secuencia. Por ejemplo, el remitente puede establecer todos los campos `CorrelId` de los mensajes de una secuencia en un valor que sea exclusivo para esa secuencia de mensajes.
   - El receptor no cambia deliberadamente el orden de recuperación, por ejemplo, especificando un `MsgId` o `CorrelId`determinado.

   Si la aplicación emisora pone los mensajes como un grupo de mensajes, los mensajes se presentan a la aplicación receptora en el orden correcto si la aplicación receptora especifica la opción MQGMO_LOGICAL_ORDER en la llamada MQGET. Para obtener más información sobre los grupos de mensajes, consulte:

   - MQMD - Campo MsgFlags
   - MQPMO_LOGICAL_ORDER
   - MQGMO_LOGICAL_ORDER

   Si el usuario está obteniendo mensajes de un grupo bajo el punto de sincronización, deben garantizar que se haya procesado todo el grupo antes de intentar finalizar la transacción.

6. Las aplicaciones deben probar el código de comentarios MQFB_QUIT en el campo `Feedback` del parámetro **`MsgDesc`** y finalizar si encuentran este valor. Consulte MQMD - Campo Feedback para obtener más información.

7. Si la cola identificada mediante `Hobj` se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT y el código de finalización de la llamada MQGET es MQCC_OK o MQCC_WARNING, el contexto asociado con el descriptor de contexto de cola `Hobj` se establece en el contexto del mensaje que se ha recuperado (a menos que se establezca la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT o MQGMO_BROWSE_MSG_UNDER_CURSOR, en cuyo caso el contexto se marca como no disponible).

   Puede utilizar el contexto guardado en una llamada MQPUT o MQPUT1 posterior especificando las opciones MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT. Esto permite que el contexto del mensaje recibido se transfiera en su totalidad o en parte a otro mensaje (por ejemplo, cuando el mensaje se reenvía a otra cola). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

8. Si incluye la opción MQGMO_CONVERT en el parámetro **GetMsgOpts** , los datos del mensaje de aplicación se convierten a la representación solicitada por la aplicación receptora, antes de que los datos se coloquen en el parámetro **Buffer** :

   - El campo `Format` de la información de control del mensaje identifica la estructura de los datos de aplicación y los campos `CodedCharSetId` y `Encoding` de la información de control del mensaje especifican su identificador y codificación de juego de caracteres.
   - La aplicación que emite la llamada MQGET especifica en los campos `CodedCharSetId` y `Encoding` del parámetro **MsgDesc** el identificador de juego de caracteres y la codificación a los que convertir los datos del mensaje de aplicación.

   Cuando es necesaria la conversión de los datos del mensaje, la conversión la realiza el propio gestor de colas o una salida escrita por el usuario, en función del valor del campo `Format` en la información de control del mensaje:

   - Los siguientes nombres de formato son formatos que convierte el gestor de colas; estos formatos se denominan formatos "incorporados":
     - MQFMT_ADMIN
     - MQFMT_CICS (solo z/OS)
     - MQFMT_COMMAND_1
     - MQFMT_COMMAND_2
     - MQFMT_DEAD_LETTER_HEADER
     - MQFMT_DIST_HEADER
     - MQFMT_EVENT versión 1
     - MQFMT_EVENT versión 2 (solo z/OS)
     - MQFMT_IMS
     - MQFMT_IMS_VAR_STRING
     - MQFMT_MD_EXTENSION
     - MQFMT_PCF
     - MQFMT_REF_MSG_HEADER
     - MQFMT_RF_HEADER
     - MQFMT_RF_HEADER_2
     - MQFMT_STRING
     - MQFMT_TRIGGER
     - MQFMT_WORK_INFO_HEADER (solo z/OS)
     - MQFMT_XMIT_Q_HEADER
   - El nombre de formato MQFMT_NONE es un valor especial que indica que la naturaleza de los datos del mensaje no está definida. Como consecuencia, el gestor de colas no intenta la conversión cuando se recupera el mensaje de la cola.

Nota: Si se especifica MQGMO_CONVERT en la llamada MQGET para un mensaje que tiene un nombre de formato de MQFMT_NONE, y el juego de caracteres o la codificación del mensaje difiere del especificado en el parámetro **MsgDesc** , el mensaje se devuelve en el parámetro **Buffer** (suponiendo que no haya otros errores), pero la llamada se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

Puede utilizar MQFMT_NONE cuando la naturaleza de los datos del mensaje significa que no requieren conversión o cuando las aplicaciones emisoras o receptoras han acordado entre ellas la forma de envío de los datos del mensaje.

- Los demás nombres de formato pasan el mensaje a una salida escrita por el usuario para su conversión. La salida tiene el mismo nombre que el formato, aparte de las adiciones específicas del entorno. Los nombres de formato especificados por el usuario no deben empezar por las letras IBM MQ.

  Consulte "salida de conversión de datos" en la página 923 para obtener más información sobre la salida de conversión de datos.

Los datos de usuario del mensaje se pueden convertir entre cualquier juego de caracteres y codificación admitidos. Sin embargo, tenga en cuenta que, si el mensaje contiene una o más estructuras de cabecera IBM MQ , el mensaje no se puede convertir de o a un juego de caracteres que tenga caracteres de doble byte o de varios bytes para cualquiera de los caracteres válidos en los nombres de cola. Si se intenta, se devuelven los códigos de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR y el mensaje sin convertir. El juego de caracteres Unicode UTF-16 es un ejemplo de este tipo de juego de caracteres.

En la devolución de la llamada MQGET, el código de razón siguiente indica que el mensaje se ha convertido correctamente:

- MQRC_NONE

El siguiente código de razón indica que el mensaje puede haberse convertido correctamente; la aplicación debe comprobar los campos `CodedCharSetId` y `Encoding` en el parámetro **MsgDesc** para averiguarlo:

- MQRC_TRUNCATED_MSG_ACCEPTED

Los demás códigos de razón indican que el mensaje no se ha convertido.

Nota: La interpretación de este código de razón es verdadera para las conversiones realizadas por una salida escrita por el usuario sólo si la salida se ajusta a las directrices de proceso descritas en "salida de conversión de datos" en la página 923.

9. Si utilice la interfaz orientada a objetos para obtener mensajes, puede optar por no especificar un almacenamiento intermedio para guardar los datos de mensajes de una llamada MQGET. Sin embargo, en las versiones de IBM MQ, anteriores a IBM WebSphere MQ 7.0, era posible que MQGET fallara con el código de razón MQRC_CONVERTED_MSG_TO_BIG, incluso cuando no se especificaba un almacenamiento intermedio. A partir de IBM WebSphere MQ 7.0, cuando obtiene un mensaje utilizando una aplicación orientada a objetos sin restringir el tamaño del almacenamiento intermedio de recepción de mensajes, la aplicación no falla con MQRC_CONVERTED_MSG_TOO_BIG y recibe el mensaje convertido. Esto se cumple en los entornos siguientes:

   - .NET, incluidas las aplicaciones totalmente gestionadas
   - C++
   - Java (IBM MQ classes for Java)

   Nota: Para todos los clientes, si el valor de `sharingConversations` es cero, el canal funciona como lo hacía antes de IBM WebSphere MQ 7.0, y el manejo de mensajes vuelve al comportamiento de IBM WebSphere MQ 6 . En esta situación, si el almacenamiento intermedio es demasiado pequeño para recibir el mensaje convertido, se devuelve el mensaje sin convertir con el código de razón MQRC_CONVERTED_MSG_TOO_BIG. Para obtener más información sobre `sharingConversations`, consulte Utilización de la compartición de conversaciones en una aplicación cliente.

10. Para los formatos incorporados, el gestor de colas puede realizar la *conversión predeterminada* de series de caracteres en el mensaje cuando se especifica la opción MQGMO_CONVERT. La conversión predeterminada permite que, para convertir datos de series de caracteres, el gestor de colas utilice un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de conjunto de caracteres real. Como consecuencia, la llamada MQGET puede realizarse con éxito con el código de terminación MQCC_OK, en vez de terminar con MQCC_WARNING y los códigos de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR.

    Nota: El resultado de utilizar un juego de caracteres aproximado para convertir datos de series de caracteres es que es posible que algunos se conviertan de forma incorrecta. Para evitarlo, en la serie utilice caracteres que sean comunes para el juego de caracteres actual y el juego de caracteres predeterminado.

    La conversión predeterminada se aplica a los datos de los mensajes de la aplicación y a los campos de tipo carácter de las estructuras MQMD y MQMDE:

    - La conversión predeterminada de los datos del mensaje de aplicación sólo se produce cuando se cumplen todas las sentencias siguientes:
      - La aplicación especifica MQGMO_CONVERT.
      - El mensaje contiene datos que deben convertirse de o a un juego de caracteres que no se admite.
      - La conversión predeterminada no estaba activada cuando se ha instalado o reiniciado el gestor de colas.
    - La conversión predeterminada de los campos de caracteres de las estructuras MQMD y MQMDE se produce según convenga, si dicha conversión predeterminada está habilitada para el gestor de colas. La conversión se realiza aunque la aplicación no especifique la opción MQGMO_CONVERT en la llamada MQGET.
11. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:
    - Si el tamaño del parámetro **`Buffer`** es menor que la longitud especificada por el parámetro **`BufferLength`** , la llamada falla con el código de razón MQRC_STORAGE_NOT_AVAILABLE.
    - El parámetro **`Buffer`** se declara como de tipo `String`. Si los datos que se van a recuperar de la cola no son de tipo `String` utilice elllamada MQGETAny en lugar de MQGET.

      La llamada MQGETAny tiene los mismos parámetros que la llamada MQGET, excepto que el parámetro **`Buffer`** se declara como de tipo `Any`, lo que permite recuperar cualquier tipo de datos. Sin embargo, esto significa que `Buffer` no se puede comprobar para asegurarse de que tiene como mínimo `BufferLength` bytes de tamaño.
12. No todas las opciones MQGET se admiten si la lectura anticipada está habilitada. En la tabla siguiente se indica qué opciones están permitidas y si se pueden modificar entre llamadas MQGET.

Tabla 548. Opciones MQGET permitidas cuando la lectura anticipada está habilitada

	Permitidas cuando la lectura anticipada está habilitada y se pueden modificar entre llamadas MQGET	**Permitido cuando la lectura anticipada está habilitada pero no se puede modificar entre llamadas MQGET** [a]	**Opciones MQGET que no están permitidas cuando la lectura anticipada está habilitada** [b]
Valores MD de MQGET	MsgId [c] CorrelId [c]	Codificación CodedCharSetId	
Opciones MQGET MQGMO	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST [d] MQGMO_BROWSE_NEXT [d] MQGMO_BROWSE_MESSAGE _UNDER_CURSOR [d]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER _CURSOR [d] MQGMO_LOCK MQGMO_UNLOCK
Valores MQGMO		MsgHandle	

a. Si estas opciones se modifican entre llamadas MQGET se devuelve el código de razón MQRC_OPTIONS_CHANGED.
b. Si estas opciones se especifican en la primera llamada MQGET, la lectura anticipada está inhabilitada. Si estas opciones se especifican en una llamada MQGET posterior, se devuelve el código de razón MQRC_OPTIONS_ERROR.
c. Las aplicaciones cliente han de tener presente que si los valores MsgId y CorrelId se modifican entre llamadas MQGET, es posible que los mensajes con los valores anteriores ya se hayan enviado al cliente y permanezcan en el almacenamiento intermedio de lectura anticipada del cliente hasta que se consuman (o se depuren automáticamente).
d. La primera llamada MQGET determina si los mensajes se han de examinar u obtener de una cola cuando la lectura anticipada está habilitada. Si la aplicación intenta utilizar una combinación de opciones de examen y de obtención, se devuelve el código de razón MQRC_OPTIONS_CHANGED.
e. MQGMO_MSG_UNDER_CURSOR no es posible con la lectura anticipada. Los mensajes se pueden examinar u obtener cuando la lectura anticipada está habilitada, pero no se puede hacer ambas cosas a la vez.

13. Las aplicaciones pueden obtener mensajes sin confirmar de forma destructiva sólo si los mensajes se han colocado en la misma unidad de trabajo local desde la que se obtienen. Las aplicaciones no pueden obtener mensajes sin confirmar de forma no destructiva.
14. Los mensajes en un cursor para examinar se pueden recuperar en una unidad de trabajo. No es posible recuperar de esta forma un mensaje sin confirmar.

Invocación en C

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

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQHOBJ   Hobj;          /* Object handle */
MQMD     MsgDesc;       /* Message descriptor */
MQGMO    GetMsgOpts;    /* Options that control the action of MQGET */
MQLONG   BufferLength;  /* Length in bytes of the Buffer area */
MQBYTE   Buffer[n];     /* Area to contain the message data */
MQLONG   DataLength;    /* Length of the message */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object handle
 01  HOBJ          PIC S9(9) BINARY.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQGET
 01  GETMSGOPTS.
     COPY CMQGMOV.
**   Length in bytes of the BUFFER area
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the message data
 01  BUFFER        PIC X(n).
**   Length of the message
```

```
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

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

Declare los parámetros como se indica a continuación:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl Hobj          fixed bin(31);  /* Object handle */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl GetMsgOpts    like MQGMO;     /* Options that control the action of
                                     MQGET */
dcl BufferLength  fixed bin(31);  /* Length in bytes of the Buffer
                                     area */
dcl Buffer        char(n);        /* Area to contain the message data */
dcl DataLength    fixed bin(31);  /* Length of the message */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
         CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN         DS       F      Connection handle
HOBJ          DS       F      Object handle
MSGDESC       CMQMDA   ,      Message descriptor
GETMSGOPTS    CMQGMOA  ,      Options that control the action of MQGET
BUFFERLENGTH  DS       F      Length in bytes of the BUFFER area
BUFFER        DS       CL(n)  Area to contain the message data
DATALENGTH    DS       F      Length of the message
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn        As Long   'Connection handle'
Dim Hobj         As Long   'Object handle'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim GetMsgOpts   As MQGMO  'Options that control the action of MQGET'
Dim BufferLength As Long   'Length in bytes of the Buffer area'
Dim Buffer       As String 'Area to contain the message data'
Dim DataLength   As Long   'Length of the message'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQINQ-Consultar atributos de objeto

La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

Son válidos los siguientes tipos de objeto:

- Gestor de colas
- Cola
- Lista de nombres
- Definición de proceso

Sintaxis

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectores*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN -entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS for CICS , se puede omitir la llamada MQCONN y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ -entrada

Este descriptor de contexto representa el objeto (de cualquier tipo) con los atributos necesarios. El descriptor de contexto debe ser devuelto por una llamada MQOPEN anterior que haya especificado la opción MQOO_INQUIRE .

SelectorCount

Tipo: MQLONG -entrada

Este es el recuento de selectores que se proporcionan en la matriz *Selectors* . Es el número de atributos que se van a devolver. Cero es un valor válido. El número máximo permitido es 256.

Selectores

Tipo: MQLONG x *SelectorCount* -entrada

Esta es una matriz de selectores de atributos **SelectorCount** ; cada selector identifica un atributo (entero o carácter) con un valor que es necesario.

Cada selector debe ser válido para el tipo de objeto que *Hobj* representa, de lo contrario la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_SELECTOR_ERROR.

En el caso especial de las colas:

- Si el selector no es válido para colas de cualquier tipo, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_SELECTOR_ERROR.
- Si el selector sólo se aplica a colas de tipos distintos al tipo del objeto, la llamada se realiza correctamente con el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE.
- Si la cola que se está consultando es una cola de clúster, los selectores que son válidos dependen de cómo se haya resuelto la cola; consulte “Notas de uso” en la página 729 para obtener más detalles.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectores MQIA_*) se devuelven en *IntAttrs* en el mismo orden en el que aparecen estos selectores en *Selectors*. Los valores de atributo que corresponden a selectores de atributo de carácter (selectores MQCA_*) se devuelven en *CharAttrs* en el mismo orden en el que se producen dichos selectores. Los selectores MQIA_* se pueden intercalar con los selectores MQCA_* ; sólo es importante el orden relativo dentro de cada tipo.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores MQIA_* residen dentro del rango de MQIA_FIRST a MQIA_LAST, y los selectores MQCA_* dentro del rango de MQCA_FIRST a MQCA_LAST.

   Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USED definen el valor más alto que acepta el gestor de colas.
2. Si todos los selectores MQIA_* aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices *Selectors* y *IntAttrs* .
3. Si el parámetro **SelectorCount** es cero, no se hace referencia a *Selectors* . En este caso, la dirección de parámetro pasada por los programas escritos en C o S/390 assembler podría ser nula.

Los atributos que se pueden consultar se listan en las tablas siguientes. Para los selectores MQCA_* , la constante que define la longitud en bytes de la serie resultante en *CharAttrs* se proporciona entre paréntesis.

Las tablas siguientes listan los selectores, por objeto, en orden alfabético, como se indica a continuación:

- Selectores de atributos de Tabla 549 en la página 716 MQINQ para colas
- Selectores de atributos de Tabla 550 en la página 719 MQINQ para listas de nombres
- Selectores de atributos de Tabla 551 en la página 719 MQINQ para definiciones de proceso
- Selectores de atributos de Tabla 552 en la página 720 MQINQ para el gestor de colas

Todos los selectores están soportados en todas las plataformas IBM MQ , excepto donde se indica en la columna **Nota** como se indica a continuación:

NOz/OS
: Soportado en todas las plataformas **excepto** z/OS

z/OS
: **solo** soportado en z/OS

Tabla 549. Selectores de atributos de MQINQ para colas

Selector	Longitud del campo	Descripción	Nota
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de reposición en cola de restitución excesivo	
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola en la que se resuelve el alias	
MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nombre de estructura de recurso de acoplamiento	z/OS
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nombre del canal de clúster emisor que utiliza esta cola como cola de transmisión.	

Tabla 549. Selectores de atributos de MQINQ para colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster	
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Lista de nombres de clúster	
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Fecha de creación de cola	
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Tiempo de creación de cola	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	El atributo personalizado para nuevas características	
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nombre cola iniciación	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de definición de proceso	
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Descripción de la cola	
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola	
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del gestor de colas remoto	
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola remota tal como se conoce en el gestor de colas remoto	
MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nombre de clase de almacenamiento	z/OS
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Datos desencadenantes	
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de transmisión	
MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de datos de contabilidad para cola	NOz/OS
MQIA_BACKOUT_THRESHOLD	MQLONG	Umbral de restituciones	
MQIA_CLWL_Q_PRIORITY	MQLONG	Prioridad de cola	
MQIA_CLWL_Q_RANK	MQLONG	Rango de cola	
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas	
MQIA_CURRENT_Q_DEPTH	MQLONG	Número de mensajes en cola	
MQIA_DEF_BIND	MQLONG	Enlace predeterminado	
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Opción de apertura para entrada predeterminada	
MQIA_DEF_PERSISTENCE	MQLONG	Persistencia de mensajes predeterminada	
MQIA_DEF_PRIORITY	MQLONG	Prioridad de mensajes predeterminada	
MQIA_DEFINITION_TYPE	MQLONG	Tipo de definición de cola	
MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución	NOz/OS

Tabla 549. Selectores de atributos de MQINQ para colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_HARDEN_GET_BACKOUT	MQLONG	Si se debe reforzar el recuento de restituciones	
MQIA_INDEX_TYPE	MQLONG	Tipo de índice mantenido para cola	z/OS
MQIA_INHIBIT_GET	MQLONG	Si se permiten operaciones get	
MQIA_INHIBIT_PUT	MQLONG	Si se permiten operaciones de colocación	
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje	
MQIA_MAX_Q_DEPTH	MQLONG	Número máximo de mensajes permitidos en la cola	
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Si la prioridad del mensaje es relevante	
MQIA_NPM_CLASS	MQLONG	Nivel de fiabilidad para mensajes no persistentes	
MQIA_OPEN_INPUT_COUNT	MQLONG	Número de llamadas MQOPEN que tienen la cola abierta para entrada	
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Número de llamadas MQOPEN que tienen la cola abierta para salida	
MQIA_PROPERTY_CONTROL	MQLONG	Atributo de control de propiedad	
MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Atributo de control para sucesos de profundidad de cola alta	NOz/OS
MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Límite alto para profundidad de cola	NOz/OS
MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Atributo de control para sucesos de profundidad de cola baja	NOz/OS
MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Límite bajo para profundidad de cola	NOz/OS
MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Atributo de control para sucesos máximos de profundidad de cola	NOz/OS
MQIA_Q_SERVICE_INTERVAL	MQLONG	Límite para intervalo de servicio de cola	NOz/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Atributo de control para sucesos de intervalo de servicio de cola	NOz/OS
MQIA_Q_TYPE	MQLONG	Tipo de cola	
MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas	z/OS
MQIA_RETENTION_INTERVAL	MQLONG	Intervalo de retención de cola	
MQIA_SCOPE	MQLONG	Ámbito de definición de cola	NOz/OS
MQIA_SHAREABILITY	MQLONG	Si la cola se puede compartir para entrada	

Tabla 549. Selectores de atributos de MQINQ para colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos estadísticos para la cola	NOz/OS
MQIA_TRIGGER_CONTROL	MQLONG	Activar control	
MQIA_TRIGGER_DEPTH	MQLONG	Profundidad de desencadenante	
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Prioridad de mensaje de umbral para desencadenantes	
MQIA_TRIGGER_TYPE	MQLONG	Tipo de desencadenante	
MQIA_USAGE	MQLONG	Utilización	

Tabla 550. Selectores de atributos de MQINQ para listas de nombres			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Descripción de lista de nombres	
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de lista de nombres	
MQIA_NAMELIST_TYPE	MQLONG	Tipo de lista de nombres	z/OS
MQCA_NAMES	MQ_Q_NAME_LENGTH x *Number of names in the list*	Nombres en la lista de nombres	
MQIA_NAME_COUNT	MQLONG	Número de nombres en la lista de nombres	
MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas	z/OS

Tabla 551. Selectores de atributos de MQINQ para definiciones de proceso			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identificador de aplicación	
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Datos de entorno	
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Descripción de la definición de proceso	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de definición de proceso	

Tabla 551. Selectores de atributos de MQINQ para definiciones de proceso (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Datos de usuario	
MQIA_APPL_TYPE	MQLONG	Tipo de aplicación	
MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas	z/OS

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de salida de definición de canal automática	
MQCA_CHINIT_SERVICE_PARM		Reservado para uso de IBM	
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Datos pasados a salida de carga de trabajo de clúster	
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de salida de carga de trabajo de clúster	
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de entrada de mandatos del sistema	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	El atributo personalizado para nuevas características	
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola de mensajes no entregados	
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de transmisión predeterminado	
MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Nombre del grupo para el escucha TCP que maneja las transmisiones de entrada para que se una el grupo de compartición de colas. El nombre se aplica cuando se utiliza Workload Manager Dynamic Domain Name Services.	z/OS
MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identificador de usuario de transferencia a colas dentro del grupo	z/OS
MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Descripción de la instalación asociada	No z/OS. NOI BM i

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nombre de la instalación asociada con el gestor de colas	No z/OS. NOIBM i
MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Vía de acceso donde está instalado el IBM MQ asociado	No z/OS. NOIBM i
MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para que las utilice el grupo de compartición de colas	z/OS
MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nombre de la LU a utilizar para las transmisiones de LU de salida 6.2 . Establezca este nombre en la misma LU que utiliza el escucha para las transmisiones de entrada	z/OS
MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Sufijo del SYS1.PARMLIB miembro APPCPM *xx*, que nombra el LUADD para este iniciador de canal	z/OS
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nombre de un gestor de colas conectado jerárquicamente que está nominado como padre de este gestor de colas	
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Descripción del gestor de colas	
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGT	Identificador de gestor de colas (H)	
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del gestor de colas local	
MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nombre del grupo de compartición de colas	z/OS
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster para el que el gestor de colas proporciona servicios de repositorio	
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que el gestor de colas proporciona servicios de repositorio	
MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Nombre del sistema TCP/IP que está utilizando	z/OS
MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Alterar temporalmente valores de contabilidad	NOz/OS

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_ACCOUNTING_INTERVAL	MQLONG	Frecuencia con la que se escriben registros de contabilidad intermedios	NOz/OS
MQIA_ACCOUNTING_MQI	MQLONG	Controla la recopilación de información de contabilidad para datos MQI	NOz/OS
MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de información de contabilidad para colas	NOz/OS
MQIA_ACTIVE_CHANNELS	MQLONG	Número máximo de canales que pueden estar activos en cualquier momento	z/OS
MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementos que se comprueban para determinar si se debe adoptar un MCA. La comprobación se realiza cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Cantidad de tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano	NOz/OS
MQIA_ADOPTNEWMCA_TYPE	MQLONG	Indica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado cuando se detecta una nueva solicitud de canal de entrada que coincide con los parámetros AdoptNewMCACheck	z/OS
MQIA_AUTHORITY_EVENT	MQLONG	Atributo de control para sucesos de autorización	NOz/OS
MQIA_BRIDGE_EVENT	MQLONG	Atributo de control para sucesos de puente IMS	z/OS
MQIA_CHANNEL_AUTO_DEF	MQLONG	Atributo de control para definición de canal automática	NOz/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Atributo de control para sucesos de definición de canal automática	NOz/OS
MQIA_CHANNEL_EVENT	MQLONG	Atributo de control para sucesos de canal	
MQIA_CHINIT_ADAPTERS	MQLONG	Número de subtareas de adaptador que se deben utilizar para procesar llamadas de IBM MQ	z/OS
MQIA_CHINIT_DISPATCHERS	MQLONG	Número de asignadores a utilizar para el iniciador de canal	z/OS
MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Indica si se debe iniciar automáticamente el rastreo de iniciador de canal	z/OS

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Tamaño del espacio de datos de rastreo (en MB) del iniciador de canal	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Longitud de carga de trabajo de clúster.	
MQIA_CLWL_MRU_CHANNELS	MQLONG	Número de canales utilizados más recientemente para el equilibrio de carga de trabajo de clúster	
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas	
MQIA_CODED_CHAR_SET_ID	MQLONG	Identificador de juego de caracteres codificado	
MQIA_COMMAND_EVENT	MQLONG	Atributo de control para sucesos de mandato	
MQIA_COMMAND_LEVEL	MQLONG	Nivel de mandatos soportado por el gestor de colas	
MQIA_CONFIGURATION_EVENT	MQLONG	Atributo de control para sucesos de configuración	NOz/OS
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Tipo de cola de transmisión predeterminada que se debe utilizar para canales de clúster emisor.	
MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución	NOz/OS
MQIA_DNS_WLM	MQLONG	Indica si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas se registra con el Gestor de carga de trabajo para Dynamic Domain Name Services	z/OS
MQIA_EXPIRY_INTERVAL	MQLONG	Intervalo entre exploraciones de mensajes caducados	z/OS
MQIA_GROUP_UR	MQLONG	Atributo de control para determinar si las unidades de recuperación GROUP están habilitadas para este gestor de colas. La disposición de unidad de recuperación GROUP sólo está disponible si el gestor de colas es miembro de un grupo de compartición de colas	z/OS
MQIA_IGQ_PUT_AUTHORITY	MQLONG	Autorización de transferencia a colas dentro del grupo	z/OS
MQIA_INHIBIT_EVENT	MQLONG	Atributo de control para sucesos de inhibición	NOz/OS
MQIA_INTRA_GROUP_queuing	MQLONG	Soporte de transferencia a colas dentro del grupo	z/OS

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_LISTENER_TIMER	MQLONG	El intervalo de tiempo (en segundos) entre IBM MQ intenta reiniciar el escucha si APPC o TCP/IP han fallado.	z/OS
MQIA_LOCAL_EVENT	MQLONG	Atributo de control para sucesos locales	NOz/OS
MQIA_LOGGER_EVENT	MQLONG	Atributo de control para sucesos de inhibición	NOz/OS
MQIA_LU62_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, utilizando el protocolo de transmisión LU 6.2	z/OS
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Intervalo de tiempo (en milisegundos) después del cual el gestor de colas puede eliminar automáticamente una marca de los mensajes de examen. **Atención:** No debe establecer este valor por debajo del valor predeterminado de 5000.	
MQIA_MAX_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales (incluidos los canales de conexión de servidor con clientes conectados)	z/OS
MQIA_MAX_HANDLES	MQLONG	Número máximo de descriptores de contexto	
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje	
MQIA_MAX_PRIORITY	MQLONG	Prioridad máxima	
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Número máximo de mensajes no confirmados dentro de una unidad de trabajo	
MQIA_OUTBOUND_PORT_MAX	MQLONG	Con MQIA_OUTBOUND_PORT_MIN, define el rango de números de puerto a utilizar al enlazar canales de salida	z/OS
MQIA_OUTBOUND_PORT_MIN	MQLONG	Con MQIA_OUTBOUND_PORT_MAX, define el rango de números de puerto a utilizar al enlazar canales de salida	z/OS
MQIA_PERFORMANCE_EVENT	MQLONG	Atributo de control para sucesos de rendimiento	NOz/OS
MQIA_PLATFORM	MQLONG	Plataforma en la que reside el gestor de colas	
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Indica si las prestaciones de seguridad de Advanced Message Security están disponibles para un gestor de colas.	

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	El número de intentos de volver a procesar un mensaje de mandato fallido bajo el punto de sincronización	
MQIA_PUBSUB_MODE	MQLONG	Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando. Las aplicaciones para publicar o suscribirse utilizando la interfaz de programación de aplicaciones requieren el motor de publicación/suscripción. Las colas supervisadas por la interfaz de publicación/suscripción en cola requieren que la interfaz de publicación/suscripción en cola esté en ejecución.	
MQIA_PUBSUB_NP_MSG	MQLONG	Si se debe descartar (o conservar) un mensaje de entrada no entregado	
MQIA_PUBSUB_NP_RESP	MQLONG	Controla el comportamiento de los mensajes de respuesta no entregados	
MQIA_PUBSUB_SYNC_PT	MQLONG	Si sólo se procesan los mensajes persistentes (o todos) bajo el punto de sincronización	
MQIA_QMGR_CFCONLOS	MQLONG	Especifica la acción que se debe realizar cuando el gestor de colas pierde la conectividad con la estructura de administración o con cualquier estructura CF con CFCONLOS establecido en ASQMGR	z/OS
MQIA_RECEIVE_TIMEOUT	MQLONG	Aproximadamente el tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo. El valor es numérico, calificado por MQIA_RECEIVE_TIMEOUT_TYPE.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Tiempo mínimo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	Aproximadamente el tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo. MQIA_RECEIVE_TIMEOUT_TYPE es el calificador aplicado a MQIA_RECEIVE_TIMEOUT.	z/OS

Tabla 552. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQIA_REMOTE_EVENT	MQLONG	Atributo de control para sucesos remotos	NOz/OS
MQIA_SECURITY_CASE	MQLONG	Caso de perfiles de seguridad	z/OS
MQIA_SSL_EVENT	MQLONG	Atributo de control para sucesos de canal	
MQIA_SSL_FIPS_REQUIRED	MQLONG	Utilizar sólo algoritmos certificados por FIPS para criptografía	
MQIA_SSL_RESET_COUNT	MQLONG	Recuento de restablecimiento de clave TLS	
MQIA_START_STOP_EVENT	MQLONG	Atributo de control para sucesos de detención de inicio	NOz/OS
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Controla la recopilación de información de supervisión de estadísticas para canales emisores de clúster	
MQIA_STATISTICS_CHANNEL	MQLONG	Controla la recopilación de datos estadísticos para canales	
MQIA_STATISTICS_INTERVAL	MQLONG	Frecuencia con la que se escriben datos de supervisión de estadísticas	NOz/OS
MQIA_STATISTICS_MQI	MQLONG	Controla la recopilación de información de supervisión de estadísticas para el gestor de colas	NOz/OS
MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos de estadísticas para colas	NOz/OS
MQIA_SYNCPOINT	MQLONG	disponibilidad de punto de sincronización	
MQIA_TCP_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, utilizando el protocolo de transmisión TCP/IP	z/OS
MQIA_TCP_KEEP_ALIVE	MQLONG	Indica si se debe utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible	z/OS
MQIA_TCP_STACK_TYPE	MQLONG	Si el iniciador de canal puede utilizar sólo el espacio de direcciones TCP/IP especificado en TCPNAME, o puede enlazarse opcionalmente a cualquier dirección TCP/IP seleccionada	z/OS
MQIA_TRACE_ROUTE_RECORDING	MQLONG	Controla el registro de la información de ruta de rastreo	z/OS
MQIA_TREE_LIFE_TIME	MQLONG	Duración de los temas no administrativos no utilizados	
MQIA_TRIGGER_INTERVAL	MQLONG	Activar intervalo	

IntAttrCount
Tipo: MQLONG -entrada

Es el número de elementos de la matriz *IntAttrs* . Cero es un valor válido.

Si IntAttrCount es como mínimo el número de selectores MQIA_* en el parámetro **Selectors** , se devuelven todos los atributos enteros solicitados.

IntAttrs
Tipo: MQLONG x *IntAttrCount* -salida

Esta es una matriz de valores de atributo de entero de *IntAttrCount* .

Los valores de atributo de entero se devuelven en el mismo orden que los selectores MQIA_* en el parámetro **Selectors** . Si la matriz contiene más elementos que el número de selectores MQIA_* , el exceso de elementos no se modificará.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve el valor específico MQIAV_NOT_APPLICABLE . Se devuelve para el elemento correspondiente en la matriz *IntAttrs* .

Si el parámetro **IntAttrCount** o **SelectorCount** es cero, no se hace referencia a *IntAttrs* . En este caso, la dirección de parámetro pasada por los programas escritos en C o S/390 assembler podría ser nula.

CharAttrLongitud
Tipo: MQLONG -entrada

Es la longitud en bytes del parámetro **CharAttrs** .

CharAttrLength debe ser como mínimo la suma de las longitudes de los atributos de caracteres solicitados (consulte Selectores). Cero es un valor válido.

CharAttrs
Tipo: MQCHAR x *CharAttrLength* -salida

Es el almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenados entre sí. La longitud del búfer se proporciona en el parámetro **CharAttrLength**.

Los atributos de carácter se devuelven en el mismo orden que los selectores MQCA_* en el parámetro **Selectors** . La longitud de cada serie de atributo es fija para cada atributo (consulte Selectores), y el valor que contiene se rellena a la derecha con espacios en blanco si es necesario. Puede proporcionar un almacenamiento intermedio mayor que el necesario para contener todos los atributos de caracteres solicitados y el relleno. Los bytes más allá del último valor de atributo devuelto no se modifican.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve una serie de caracteres que consta totalmente de asteriscos (*). El asterisco se devuelve como el valor de ese atributo en *CharAttrs*.

Si el parámetro *CharAttrLength* o **SelectorCount** es cero, no se hace referencia a *CharAttrs* . En este caso, la dirección de parámetro pasada por los programas escritos en C o S/390 assembler podría ser nula.

CompCode
Tipo: MQLONG -salida

El código de terminación:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es `MQCC_OK`:

MQRC_NONE
(`0, X'000'`) No hay razón para informar.

Si *CompCode* es `MQCC_WARNING`:

MQRC_CHAR_ATTRS_TOO_SHORT
(`2008, X'7D8'`) No se permite espacio suficiente para los atributos de tipo carácter.

MQRC_INT_ATTR_COUNT_TOO_SMALL
(`2022, X'7E6'`) No se permite suficiente espacio para atributos enteros.

MQRC_SELECTOR_NOT_FOR_TYPE
(`2068, X'814'`) El selector no es aplicable al tipo de cola.

Si *CompCode* es `MQCC_FAILED`:

MQRC_ADAPTER_NOT_AVAILABLE
(`2204, X'89C'`) Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(`2130, X'852'`) No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(`2374, X'946'`) Ha fallado la salida de API.

MQRC_API_EXIT_LOAD_ERROR
(`2183, X'887'`) No se puede cargar la salida de API.

MQRC_ASID_MISMATCH
(`2157, X'86D'`) Los ASID primario y de inicio difieren.

MQRC_CALL_IN_PROGRESS
(`2219, X'8AB'`) Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CF_STRUC_FAILED
(`2373, X'945'`) La estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(`2346, X'92A'`) Estructura de recurso de acoplamiento en uso.

MQRC_CHAR_ATTR_LENGTH_ERROR
(`2006, X'7D6'`) La longitud de los atributos de caracteres no es válida.

MQRC_CHAR_ATTRS_ERROR
(`2007, X'7D7'`) Serie de atributos de tipo carácter no válida.

MQRC_CICS_WAIT_FAILED
(`2140, X'85C'`) Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(`2009, X'7D9'`) Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(`2217, X'8A9'`) No autorizado para la conexión.

MQRC_CONNECTION_STOPPING
(`2203, X'89B'`) Se está cerrando la conexión.

MQRC_HCONN_ERROR
(`2018, X'7E2'`) El descriptor de conexión no es válido.

MQRC_HOBJ_ERROR
(`2019, X'7E3'`) Descriptor de contexto de objeto no válido.

MQRC_INT_ATTR_COUNT_ERROR
(`2021, X'7E5'`) Recuento de atributos enteros no válido.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') La matriz de atributos enteros no es válida.

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') Cola no abierta para consulta.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definición de objeto cambiada desde su apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objeto dañado.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos de conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') Cola suprimida.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestor de colas no disponible para la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') Concluyendo el gestor de colas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay suficientes recursos del sistema disponibles.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR
(2067, X'813') El selector de atributos no es válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Recuento de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Llamada suprimida por el programa de salida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos; consulte Mensajes y códigos de razón

Notas de uso

1. Los valores devueltos son una instantánea de los atributos seleccionados. No hay ninguna garantía de que los atributos permanezcan iguales antes de que la aplicación pueda actuar sobre los valores devueltos.
2. Al abrir una cola modelo, se crea una cola local dinámica. Se crea una cola local dinámica incluso si abre la cola modelo para consultar sus atributos.

   Los atributos de la cola dinámica son en gran medida los mismos que los atributos de la cola modelo en el momento en que se crea la cola dinámica. Si luego utiliza la llamada MQINQ en esta cola, el gestor de colas devuelve los atributos de la cola dinámica y no los atributos de la cola modelo. Consulte Tabla 561 en la página 850 para obtener detalles sobre qué atributos de la cola modelo hereda la cola dinámica.
3. Si el objeto que se está consultando es una cola alias, los valores de atributo devueltos por la llamada MQINQ son los atributos de la cola alias. No son los atributos de la cola base o tema en el que se resuelve el alias.
4. Si el objeto que se está consultando es una cola de clúster, los atributos que se pueden consultar dependen de cómo se abra la cola:

- Puede abrir una cola de clúster para realizar consultas más una o más de las operaciones de entrada, examinar o establecer. Para ello, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria. En este caso, los atributos que se pueden consultar son los atributos que son válidos para las colas locales.

  Si la cola de clúster está abierta para consultas sin entrada, examen o conjunto especificado, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068) si intenta consultar atributos que son válidos sólo para colas locales y no para colas de clúster.
- Puede abrir una cola de clúster para realizar consultas al pasar el nombre del gestor de colas base del gestor de colas conectado.

  Para ello, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria. Si no se pasa el gestor de colas base, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068) si intenta consultar atributos que son válidos sólo para colas locales y no para colas de clúster
- Si la cola de clúster se abre sólo para consultas, o para consultas y salidas, sólo se pueden consultar los atributos listados. El atributo **QType** tiene el valor MQQT_CLUSTER en este caso:
  - MQCA_Q_DESC
  - MQCA_Q_NAME
  - MQIA_DEF_BIND
  - MQIA_DEF_PERSISTENCE
  - MQIA_DEF_PRIORITY
  - MQIA_INHIBIT_PUT
  - MQIA_Q_TYPE

  Puede abrir la cola de clúster sin ningún enlace fijo. Puede abrirlo con MQOO_BIND_NOT_FIXED especificado en la llamada MQOPEN . De forma alternativa, especifique MQOO_BIND_AS_Q_DEFy establezca el atributo **DefBind** de la cola en MQBND_BIND_NOT_FIXED. Si abre una cola de clúster sin ningún enlace fijo, las llamadas MQINQ sucesivas para la cola pueden consultar distintas instancias de la cola de clúster. Sin embargo, es típico que todas las instancias tengan los mismos valores de atributo.
- Se puede definir un objeto de cola alias para un clúster. Puesto que TARGTYPE y TARGET no son atributos de clúster, el proceso que realiza un proceso MQOPEN en la cola alias no reconoce el objeto en el que se resuelve el alias.

  Durante el MQOPENinicial, la cola alias se resuelve en un gestor de colas y una cola del clúster. La resolución de nombres tiene lugar de nuevo en el gestor de colas remoto y es aquí donde se resuelve el TARGTPYE de la cola alias.

  Si la cola alias se resuelve en un alias de tema, la publicación de mensajes colocados en la cola alias tiene lugar en este gestor de colas remoto.

Consulte Colas de clúster

5. Es posible que desee consultar una serie de atributos y, a continuación, establecer algunos de ellos utilizando la llamada MQSET . Para programar consultas y establecer de forma eficiente, coloque los atributos que se van a establecer al principio de las matrices de selector. Si lo hace, se pueden utilizar las mismas matrices con recuentos reducidos para MQSET.
6. Si se produce más de una de las situaciones de aviso (consulte el parámetro **CompCode**), el código de razón devuelto es el primero de la lista siguiente que se aplica:
   a. MQRC_SELECTOR_NOT_FOR_TYPE
   b. MQRC_INT_ATTR_COUNT_TOO_SMALL
   c. MQRC_CHAR_ATTRS_TOO_SHORT
7. El tema siguiente tiene información sobre los atributos de objeto:

- “Atributos para colas” en la página 848
- “Atributos de las listas de nombres” en la página 883
- “Atributos de las definiciones de proceso” en la página 885
- “Atributos para el gestor de colas” en la página 810

Invocación en C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
MQLONG   Selectors[n];    /* Array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
MQLONG   IntAttrs[n];     /* Array of integer attributes */
MQLONG   CharAttrLength;  /* Length of character attributes buffer */
MQCHAR   CharAttrs[n];    /* Character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                        INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                        CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN            PIC S9(9) BINARY.
**   Object handle
 01  HOBJ             PIC S9(9) BINARY.
**   Count of selectors
 01  SELECTORCOUNT    PIC S9(9) BINARY.
**   Array of attribute selectors
 01  SELECTORS-TABLE.
  02  SELECTORS        PIC S9(9) BINARY OCCURS n TIMES.
**   Count of integer attributes
 01  INTATTRCOUNT     PIC S9(9) BINARY.
**   Array of integer attributes
 01  INTATTRS-TABLE.
  02  INTATTRS         PIC S9(9) BINARY OCCURS n TIMES.
**   Length of character attributes buffer
 01  CHARATTRLENGTH   PIC S9(9) BINARY.
**   Character attributes
 01  CHARATTRS        PIC X(n).
**   Completion code
 01  COMPCODE         PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON           PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn           fixed bin(31);  /* Connection handle */
```

```
dcl Hobj            fixed bin(31);  /* Object handle */
dcl SelectorCount   fixed bin(31);  /* Count of selectors */
dcl Selectors(n)    fixed bin(31);  /* Array of attribute selectors */
dcl IntAttrCount    fixed bin(31);  /* Count of integer attributes */
dcl IntAttrs(n)     fixed bin(31);  /* Array of integer attributes */
dcl CharAttrLength  fixed bin(31);  /* Length of character attributes
                                       buffer */
dcl CharAttrs       char(n);        /* Character attributes */
dcl CompCode        fixed bin(31);  /* Completion code */
dcl Reason          fixed bin(31);  /* Reason code qualifying
                                       CompCode */
```

Invocación en ensamblador de alto nivel

```
         CALL MQINQ,(HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,  X
               INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN           DS  F      Connection handle
HOBJ            DS  F      Object handle
SELECTORCOUNT   DS  F      Count of selectors
SELECTORS       DS  (n)F   Array of attribute selectors
INTATTRCOUNT    DS  F      Count of integer attributes
INTATTRS        DS  (n)F   Array of integer attributes
CHARATTRLENGTH  DS  F      Length of character attributes buffer
CHARATTRS       DS  CL(n)  Character attributes
COMPCODE        DS  F      Completion code
REASON          DS  F      Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn          As Long   'Connection handle'
Dim Hobj           As Long   'Object handle'
Dim SelectorCount  As Long   'Count of selectors'
Dim Selectors      As Long   'Array of attribute selectors'
Dim IntAttrCount   As Long   'Count of integer attributes'
Dim IntAttrs       As Long   'Array of integer attributes'
Dim CharAttrLength As Long   'Length of character attributes buffer'
Dim CharAttrs      As String 'Character attributes'
Dim CompCode       As Long   'Completion code'
Dim Reason         As Long   'Reason code qualifying CompCode'
```

MQINQMP-Consultar propiedad de mensaje

La llamada MQINQMP devuelve el valor de una propiedad de un mensaje.

Sintaxis

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el manejador de mensajes se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra consultando una propiedad del manejador de mensajes, de lo contrario, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg
Tipo: MQHMSG-entrada

Este es el manejador de mensajes que se debe consultar. El valor ha sido devuelto por una llamada **MQCRTMH** anterior.

InqPropOpts
Tipo: MQIMPO-entrada/salida

Consulte el tipo de datos MQIMPO para obtener más detalles.

Nombre
Tipo: MQCHARV-entrada/salida

El nombre de la propiedad que se va a consultar.

Si no se puede encontrar ninguna propiedad con este nombre, la llamada falla con la razón MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar el signo de porcentaje de carácter comodín (%) al final del nombre de propiedad. El comodín coincide con cero o más caracteres, incluido el carácter de punto (.). Esto permite a una aplicación consultar el valor de muchas propiedades. Llame a MQINQMP con la opción MQIMPO_INQ_FIRST para obtener la primera propiedad coincidente y de nuevo con la opción MQIMPO_INQ_NEXT para obtener la siguiente propiedad coincidente. Cuando no hay más propiedades coincidentes disponibles, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE. Si el campo *ReturnedName* de la estructura InqPropOpts se inicializa con una dirección o un desplazamiento para el nombre devuelto de la propiedad, esto se completa al devolver MQINQMP con el nombre de la propiedad que ha coincidido. Si el campo *VSBufSize* del *ReturnedName* en la estructura de Opts InqPropes menor que la longitud del nombre de propiedad devuelto, el código de terminación se establece en MQCC_FAILED con la razón MQRC_PROPERTY_NAME_TOO_BIG.

Las propiedades que tienen sinónimos conocidos se devuelven de la siguiente manera:

1. Propiedades con el prefijo "mqps." se devuelven como nombre de propiedad IBM MQ . Por ejemplo, "MQTopicString" es el nombre devuelto en lugar de "mqps.Top"
2. Propiedades con el prefijo "jms." o "mcd." se devuelven como el nombre de campo de cabecera JMS , por ejemplo, "JMSExpiration" es el nombre devuelto en lugar de "jms.Exp".
3. Propiedades con el prefijo "usr." se devuelven sin ese prefijo, por ejemplo, se devuelve "Color" en lugar de "usr.Color".

Las propiedades con sinónimos sólo se devuelven una vez.

En el lenguaje de programación C, las variables de macro siguientes se definen para consultar todas las propiedades y, a continuación, todas las propiedades que empiezan por "usr.":

MQPROP_INQUIRE_ALL
Consultar todas las propiedades del mensaje.

MQPROP_INQUIRE_ALL se puede utilizar de la siguiente manera:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR
Consultar todas las propiedades del mensaje que inician "usr.". El nombre devuelto se devuelve sin el "usr." .

Si se especifica MQIMP_INQ_NEXT pero el nombre ha cambiado desde la llamada anterior o ésta es la primera llamada, entonces MQIMPO_INQ_FIRST está implícito.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PropDesc
Tipo: MQPD-salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluido lo que sucede si la propiedad no está soportada, a qué contexto de mensaje pertenece la propiedad y en qué mensajes debe copiarse la propiedad. Consulte MQPD para obtener detalles de esta estructura.

Tipo
Tipo: MQLONG-entrada/salida

Al volver de la llamada MQINQMP, este parámetro se establece en el tipo de datos *Valor*. El tipo de datos puede ser cualquiera de los siguientes:

MQTYPE_BOOLEAN
Un booleano.

MQTYPE_BYTE_STRING
una serie de bytes.

MQTYPE_INT8
Un entero con signo de 8 bits.

MQTYPE_INT16
Un entero con signo de 16 bits.

MQTYPE_INT32
Un entero con signo de 32 dígitos.

MQTYPE_INT64
Un entero con signo de 64 bits.

MQTYPE_FLOAT32
Un número de coma flotante de 32 bits.

MQTYPE_FLOAT64
Un número de coma flotante de 64 bits.

MQTYPE_STRING
Una serie de caracteres.

MQTYPE_NULL
La propiedad existe pero tiene un valor nulo.

Si el tipo de datos del valor de propiedad no se reconoce, se devuelve MQTYPE_STRING y se coloca una representación de serie del valor en el área *Valor* . Se puede encontrar una representación de serie del tipo de datos en el campo *TypeString* del parámetro *InqPropOpts* . Se devuelve un código de finalización de aviso con la razón MQRC_PROP_TYPE_NOT_SUPPORTED.

Además, si se especifica la opción MQIMPO_CONVERT_TYPE, se solicita la conversión del valor de propiedad. Utilice *Tipo* como entrada para especificar el tipo de datos con el que desea que se devuelva la propiedad. Consulte la descripción de la opción MQIMPO_CONVERT_TYPE de la estructura MQIMPO para obtener detalles de la conversión de tipos de datos.

Si no solicita la conversión de tipo, puede utilizar el valor siguiente en la entrada:

MQTYPE_AS_SET
El valor de la propiedad se devuelve sin convertir su tipo de datos.

ValueLength
Tipo: MQLONG - entrada

Longitud en bytes del área Valor. Especifique cero para las propiedades para las que no necesita el valor devuelto. Estas podrían ser propiedades diseñadas por una aplicación para tener un valor nulo o

una serie vacía. Especifique también cero si se ha especificado la opción MQIMPO_QUERY_LENGTH ; en este caso, no se devuelve ningún valor.

Valor
Tipo: MQBYTEx *ValueLength* -salida

Este es el área que contiene el valor de propiedad consultado. El almacenamiento intermedio debe estar alineado en un límite adecuado para el valor que se devuelve. Si no lo hace, puede producirse un error cuando se accede al valor más tarde.

Si *ValueLength* es menor que la longitud del valor de propiedad, la mayor parte posible del valor de propiedad se mueve a *Valor* y la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_VALUE_TOO_BIG.

El juego de caracteres de los datos en *Valor* se proporciona mediante el campo ReturnedCCSID en el parámetro InqPropOpts. La codificación de los datos en *Valor* se proporciona mediante el campo ReturnedEncoding en el parámetro InqPropOpts.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro *ValueLength* es cero, no se hace referencia al *Valor* y su valor pasado por los programas escritos en C o System/390 assembler puede ser nulo.

DataLength
Tipo: MQLONG - salida

Es la longitud en bytes del valor de propiedad real tal como se devuelve en el área *Valor* .

Si *DataLength* es menor que la longitud del valor de propiedad, *DataLength* se sigue rellenando a la devolución de la llamada MQINQMP. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad y, a continuación, volver a emitir la llamada con un almacenamiento intermedio del tamaño adecuado.

También se pueden devolver los valores siguientes.

Si el parámetro *Tipo* se establece en MQTYPE_STRING o MQTYPE_BYTE_STRING:

MQVL_EMPTY_STRING
La propiedad existe pero no contiene caracteres ni bytes.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_PROP_NAME_NOT_CONVERT
(2492, X'09BC') No se ha convertido el nombre de propiedad devuelto.

MQRC_PROP_VALUE_NOT_CONVERTED
(2466, X'09A2') Valor de propiedad no convertido.

MQRC_PROP_TYPE_NOT_SUPPORTED
(2467, X'09A3') El tipo de datos de propiedad no está soportado.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'086D') Los ASID primario y de inicio difieren.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de valor no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') El parámetro de longitud de datos no es válido.

MQRC_IMPO_ERROR
(2464, X'09A0') Consultar estructura de opciones de propiedad de mensaje no válida.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07F8') Opciones no válidas o no coherentes.

MQRC_PD_ERROR
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

MQRC_PROP_CONV_NOT_SUPPORTED
(2470, X'09A6') La conversión del tipo de datos real al solicitado no está soportada.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

MQRC_PROPERTY_NAME_TOO_BIG
(2465, X'09A1') Nombre de propiedad demasiado grande para el almacenamiento intermedio de nombre devuelto.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7) Propiedad no disponible.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Valor de propiedad demasiado grande para el área Valor.

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

ERROR_TIPO_PROPIEDAD_MQRC
(2473, X'09A9') Tipo de propiedad solicitado no válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'0871 ') Almacenamiento insuficiente disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Invocación en C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;       /* Connection handle */
MQHMSG  Hmsg;        /* Message handle */
MQIMPO  InqPropOpts; /* Options that control the action of MQINQMP */
MQCHARV Name;        /* Property name */
MQPD    PropDesc;    /* Property descriptor */
MQLONG  Type;        /* Property data type */
MQLONG  ValueLength; /* Length in bytes of the Value area */
MQBYTE  Value[n];    /* Area to contain the property value */
MQLONG  DataLength;  /* Length of the property value */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDESC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN       PIC S9(9) BINARY.
**   Message handle
 01  HMSG            PIC S9(18) BINARY.
**   Options that control the action of MQINQMP
 01  INQMSGOPTS.
     COPY CMQIMPOV.
**   Property name
 01  NAME.
     COPY CMQCHRVV.
**   Property descriptor
 01  PROPDESC.
     COPY CMQPDV.
**   Property data type
 01  TYPE        PIC S9(9) BINARY.
**   Length in bytes of the VALUE area
 01  VALUELENGTH PIC S9(9) BINARY.
**   Area to contain the property value
 01  VALUE       PIC X(n).
**   Length of the property value
 01  DATALENGTH  PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE    PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON      PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl InqPropOpts like MQIMPO;    /* Options that control the action of MQINQMP */
dcl Name        like MQCHARV;   /* Property name */
dcl PropDesc    like MQPD;      /* Property descriptor */
dcl Type        fixed bin (31); /* Property data type */
dcl ValueLength fixed bin (31); /* Length in bytes of the Value area */
dcl Value       char (n);       /* Area to contain the property value */
dcl DataLength  fixed bin (31); /* Length of the property value */
dcl CompCode    fixed bin (31); /* Completion code */
dcl Reason      fixed bin (31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQINQMP,(HCONN,HMSG,INQMSGOPTS,NAME,PROPDESC,TYPE,
VALUELENGTH,VALUE,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
INQMSGOPTS   CMQIMPOA   ,     Options that control the action of MQINQMP
NAME         CMQCHRVA   ,     Property name
PROPDESC     CMQPDA     ,     Property descriptor
TYPE         DS         F     Property data type
VALUELENGTH  DS         F     Length in bytes of the VALUE area
VALUE        DS         CL(n) Area to contain the property value
DATALENGTH   DS         F     Length of the property value
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQMHBUF-Convertir descriptor de mensaje en almacenamiento intermedio

La llamada MQMHBUF convierte un manejador de mensajes en un almacenamiento intermedio y es el inverso de la llamada MQBUFMH.

Sintaxis

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Nombre*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG-entrada

Este es el descriptor de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

MsgHBufOpts

Tipo: MQMHBO-entrada

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los almacenamientos intermedios a partir de los manejadores de mensajes.

Para obtener más detalles, consulte “MQMHBO-Descriptor de mensaje para opciones de almacenamiento intermedio” en la página 481.

Nombre
Tipo: MQCHARV-entrada

El nombre de la propiedad o propiedades que se van a poner en el almacenamiento intermedio.

Si no se encuentra ninguna propiedad que coincida con el nombre, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar un comodín para colocar más de una propiedad en el almacenamiento intermedio. Para ello, utilice el carácter comodín '%' al final del nombre de propiedad. Este comodín coincide con cero o más caracteres, incluyendo '.' .

En el lenguaje de programación C, se definen las siguientes variables de macro para consultar todas las propiedades y todas las propiedades que empiezan por 'usr':

MQPROP_INQUIRE_ALL
Colocar todas las propiedades del mensaje en el almacenamiento intermedio

MQPROP_INQUIRE_ALL_USR
Ponga todas las propiedades del mensaje que empiezan con los caracteres 'usr.' en el almacenamiento intermedio.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

MsgDesc
Tipo: MQMD - entrada/salida

La estructura *MsgDesc* describe el contenido del área de almacenamiento intermedio.

En la salida, los campos *Encoding*, *CodedCharSetId* y *Format* se establecen para describir correctamente la codificación, el identificador de juego de caracteres y el formato de los datos en el área de almacenamiento intermedio tal como los graba la llamada.

Los datos de esta estructura están en el juego de caracteres y la codificación de la aplicación.

BufferLength
Tipo: MQLONG - entrada

BufferLength es la longitud del área de almacenamiento intermedio, en bytes.

Almacenamiento intermedio
Tipo: MQBYTExBufferLength - salida

Buffer define el área que contendrá las propiedades del mensaje. Debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en *Buffer*, MQMHBUF falla con MQRC_PROPERTY_VALUE_TOO_BIG.

El contenido del almacenamiento intermedio puede cambiar incluso si la llamada falla.

DataLength
Tipo: MQLONG - salida

DataLength es la longitud, en bytes, de las propiedades devueltas en el almacenamiento intermedio. Si el valor es cero, ninguna propiedad coincide con el valor proporcionado en *Name* y la llamada falla con el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en el almacenamiento intermedio, la llamada MQMHBUF falla con MQRC_PROPERTY_VALUE_TOO_BIG, pero se sigue entrando un valor en *DataLength*. Esto permite a la aplicación determinar el tamaño

del almacenamiento intermedio necesario para acomodar las propiedades y, a continuación, volver a emitir la llamada con el *BufferLength*necesario.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_MHBO_ERROR
(2501, X'095C') El descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio no es válido.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') El parámetro de longitud de datos no es válido.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MD_ERROR
(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') El nombre de propiedad no es válido.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Propiedad no disponible.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') El valor BufferLength es demasiado pequeño para contener las propiedades especificadas.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Invocación en C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;        /* Connection handle */
MQHMSG  Hmsg;         /* Message handle */
MQMHBO  MsgHBufOpts;  /* Options that control the action of MQMHBUF */
MQCHARV Name;         /* Property name */
MQMD    MsgDesc;      /* Message descriptor */
MQLONG  BufferLength; /* Length in bytes of the Buffer area */
MQBYTE  Buffer[n];    /* Area to contain the properties */
MQLONG  DataLength;   /* Length of the properties */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

Notas de uso

MQMHBUF convierte un descriptor de mensaje en un almacenamiento intermedio.

Puede utilizarlo con una salida de API MQGET para acceder a determinadas propiedades, utilizando las API de propiedades de mensajes y, a continuación, volver a pasarlas en un almacenamiento intermedio a una aplicación diseñada para utilizar cabeceras MQRFH2 en lugar de manejadores de mensajes.

Esta llamada es la inversa de la llamada MQBUFMH, que puede utilizar para analizar propiedades de mensaje de un almacenamiento intermedio en un descriptor de mensaje.

Invocación en COBOL

```
     CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                          BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Message handle
 01  HMSG          PIC S9(18) BINARY.
**   Options that control the action of MQMHBUF
 01  MSGHBUFOPTS.
     COPY CMQMHBOV.
**   Property name
 01  NAME
     COPY CMQCHRVV.
**   Message descriptor
 01  MSGDESC
     COPY CMQMDV.
**   Length in bytes of the Buffer area */
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the properties
 01  BUFFER        PIC X(n).
**   Length of the properties
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
```

```
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Hmsg         fixed bin(63); /* Message handle */
dcl MsgHBufOpts  like MQMHBO;   /* Options that control the action of MQMHBUF */
dcl Name         like MQCHARV;  /* Property name */
dcl MsgDesc      like MQMD;     /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer       char(n);       /* Area to contain the properties */
dcl DataLength   fixed bin(31); /* Length of the properties */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQMHBUF,(HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN        DS       F      Connection handle
HMSG         DS       D      Message handle
MSGHBUFOPTS  CMQMHBOA ,      Options that control the action of MQMHBUF
NAME         CMQCHRVA ,      Property name
MSGDESC      CMQMDA   ,      Message descriptor
BUFFERLENGTH DS       F      Length in bytes of the BUFFER area
BUFFER       DS       CL(n)  Area to contain the properties
DATALENGTH   DS       F      Length of the properties
COMPCODE     DS       F      Completion code
REASON       DS       F      Reason code qualifying COMPCODE
```

MQOPEN-Abrir objeto

La llamada MQOPEN establece el acceso a un objeto.

Son válidos los siguientes tipos de objeto:

- Cola (incluidas las listas de distribución)
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Sintaxis

MQOPEN (*Hconn*, *ObjDesc*, *Options*, *Hobj*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de Hconn ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

z/OS En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

ObjDesc
Tipo: MQOD-entrada/salida

Esta es una estructura que identifica el objeto que se va a abrir; consulte "MQOD-Descriptor de objeto" en la página 483 para obtener más detalles.

Si el campo `ObjectName` del parámetro **ObjDesc** es el nombre de una cola modelo, una cola local dinámica se crea con los atributos de la cola modelo; esto sucede independientemente de las opciones que especifique en el parámetro **Options** . Las operaciones posteriores que utilizan `Hobj` devueltas por la llamada MQOPEN se realizan en la nueva cola dinámica y no en la cola modelo. Esto es cierto incluso para las llamadas MQINQ y MQSET. El nombre de la cola modelo en el parámetro **ObjDesc** se sustituye por el nombre de la cola dinámica creada. El tipo de la cola dinámica viene determinado por el valor del atributo **DefinitionType** de la cola modelo (consulte "Atributos para colas" en la página 848). Para obtener información sobre las opciones de cierre aplicables a las colas dinámicas, consulte la descripción de la llamada MQCLOSE.

Opciones
Tipo: MQLONG - entrada

Debe especificar al menos una de las opciones siguientes:

- MQOO_BROWSE
- MQOO_INPUT_ * (sólo uno de estos)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_ * (sólo uno de estos)

Consulte la tabla siguiente para obtener detalles de estas opciones; se pueden especificar otras opciones según sea necesario. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit). Las combinaciones que no son válidas se anotan; todas las demás combinaciones son válidas. Sólo se permiten las opciones que son aplicables al tipo de objeto especificado por `ObjDesc` .

Tabla 553. Opciones MQOPEN válidas para colas y temas

Opción	Alias [1]	Local y modelo	Remotos	Clúster no local	Lista de distribución	Tema
MQOO_INPUT_AS_Q_DEF	Sí	Sí	No	No	No	No
MQOO_INPUT_SHARED	Sí	Sí	No	No	No	No
MQOO_INPUT_EXCLUSIVE	Sí	Sí	No	No	No	No
MQOO_OUTPUT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_BROWSE	Sí	Sí	No	No	No	No
MQOO_CO_OP	Sí	Sí	No	No	No	No
MQOO_INQUIRE	Sí	Sí	[2]	Sí	No	No
MQOO_SET	Sí	Sí	[2]	No	No	No
MQOO_BIND_ON_OPEN [3]	Sí	Sí	Sí	Sí	Sí	No

Tabla 553. Opciones MQOPEN válidas para colas y temas (continuación)						
Opción	**Alias** [1]	**Local y modelo**	**Remotos**	**Clúster no local**	**Lista de distribución**	**Tema**
MQOO_BIND_NOT_FIXED [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_BIND_ON_GROUP [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_BIND_AS_Q_DEF [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_SAVE_ALL_CONTEXT	Sí	Sí	No	No	No	No
MQOO_PASS_IDENTITY_CONTEXT	Sí	Sí	Sí	Sí	Sí	[4]
MQOO_PASS_ALL_CONTEXT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_SET_IDENTITY_CONTEXT	Sí	Sí	Sí	Sí	Sí	[4]
MQOO_SET_ALL_CONTEXT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_NO_READ_AHEAD	Sí	Sí	No	No	No	No
MQOO_READ_AHEAD	Sí	Sí	No	No	No	No
MQOO_READ_AHEAD_AS_Q_DEF	Sí	Sí	No	No	No	No
MQOO_ALTERNATE_USER_AUTHORITY	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_FAIL_IF_QUIESCING	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_RESOLVE_LOCAL_Q	Sí	Sí	Sí	Sí	No	No
MQOO_RESOLVE_LOCAL_TOPIC	No	No	No	No	No	Sí
MQOO_NO_MULTICAST	No	No	No	No	No	Sí

Notas:

1. La validez de las opciones para los alias depende de la validez de la opción para la cola en la que se resuelve el alias.
2. Esta opción sólo es válida para la definición local de una cola remota.
3. Esta opción se puede especificar para cualquier tipo de cola, pero se ignora si la cola no es una cola de clúster. Sin embargo, el atributo de cola **DefBind** altera temporalmente la cola base incluso cuando la cola alias no está en un clúster.
4. Estos atributos se pueden utilizar con un tema, pero sólo afectan al contexto establecido para el mensaje retenido, no a los campos de contexto enviados a cualquier suscriptor.

Opciones de acceso: las opciones siguientes controlan el tipo de operaciones que se pueden realizar en el objeto:

MQOO_INPUT_AS_Q_DEF
Abra la cola para obtener mensajes utilizando el valor predeterminado definido por la cola.

La cola se abre para su uso con las llamadas MQGET posteriores. El tipo de acceso es compartido o exclusivo, en función del valor del atributo de cola **DefInputOpenOption** ; consulte "Atributos para colas" en la página 848 para obtener más detalles.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INPUT_SHARED
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INPUT_EXCLUSIVE

Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente por esta u otra aplicación para cualquier tipo de entrada (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_OUTPUT

Abra la cola para transferir mensajes, o un tema o serie de tema para publicar mensajes.

La cola o tema se abre para su uso con llamadas MQPUT posteriores.

Una llamada MQOPEN con esta opción puede ser satisfactoria incluso si el atributo de cola **InhibitPut** se establece en MQQA_PUT_INITED (aunque las llamadas MQPUT posteriores fallan mientras el atributo se establece en este valor).

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución y los temas.

Las siguientes notas se aplican a estas opciones:

- Sólo se puede especificar una de estas opciones.
- Una llamada MQOPEN con una de estas opciones puede ser satisfactoria incluso si el atributo de cola **InhibitGet** se establece en MQQA_GET_INSITED (aunque las llamadas MQGET posteriores fallan mientras el atributo se establece en este valor).
- Si la cola se define como no compartible (es decir, el atributo de cola **Shareability** tiene el valor MQQA_NOT_SHAREABLE), los intentos de abrir la cola para acceso compartido se tratan como intentos de abrir la cola con acceso exclusivo.
- Si se abre una cola alias con una de estas opciones, la prueba de uso exclusivo (o si otra aplicación tiene uso exclusivo) se realiza en la cola base en la que se resuelve el alias.
- Estas opciones no son válidas si **ObjectQMgrName** es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

MQOO_BROWSE

Abra la cola para examinar los mensajes.

La cola se abre para su uso con llamadas MQGET posteriores con una de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Esto está permitido incluso si la cola está abierta actualmente para MQOO_INPUT_EXCLUSIVE. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor para examinar y lo sitúa lógicamente antes del primer mensaje en la cola; consulte Campo MQGMO-Options para obtener más información.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Tampoco es válido si `ObjectQMgrName` es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

MQOO_CO_OP

Abierto como miembro cooperante del conjunto de descriptores de contexto.

Esta opción sólo es válida con la opción MQOO_BROWSE. Si se especifica sin MQOO_BROWSE, MQOPEN devuelve con MQRC_OPTIONS_ERROR.

El descriptor de contexto devuelto se considera miembro de un conjunto de descriptores de contexto cooperantes para llamadas MQGET posteriores con una de las opciones siguientes:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INQUIRE
Abrir objeto para consultar atributos.

La cola, la lista de nombres, la definición de proceso o el gestor de colas se abre para su uso con llamadas MQINQ posteriores.

Esta opción es válida para todos los tipos de objetos que no sean listas de distribución. No es válido si `ObjectQMgrName` es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para la asignación de alias de gestor de colas es el nombre del gestor de colas local.

MQOO_SET
Abra la cola para establecer atributos.

La cola se abre para utilizarla con las llamadas MQSET posteriores.

Esta opción es válida para todos los tipos de cola que no sean listas de distribución. No es válido si `ObjectQMgrName` es el nombre de una definición local de una cola remota; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

Opciones de enlace: Las opciones siguientes se aplican cuando el objeto que se está abriendo es una cola de clúster; estas opciones controlan el enlace del descriptor de contexto de cola a una instancia de la cola de clúster:

MQOO_BIND_ON_OPEN
El gestor de colas local enlaza el descriptor de contexto de cola con una instancia de la cola de destino cuando se abre la cola. Como resultado, todos los mensajes colocados utilizando este descriptor de contexto se envían a la misma instancia de la cola de destino y por la misma ruta.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_NOT_FIXED
Esto detiene el gestor de colas local que enlaza el descriptor de contexto de cola con una instancia de la cola de destino. Como resultado, las llamadas MQPUT sucesivas que utilizan este manejador envían los mensajes a distintas instancias de la cola de destino, o a la misma instancia pero por rutas diferentes. También permite que el gestor de colas local, un gestor de colas remoto o un agente de canal de mensajes (MCA) cambien la instancia seleccionada más tarde, según las condiciones de red.

Nota: Las aplicaciones cliente y servidor que necesitan intercambiar una serie de mensajes para completar una transacción no deben utilizar MQOO_BIND_NOT_FIXED (o MQOO_BIND_AS_Q_DEF cuando `DefBind` tiene el valor MQBND_BIND_NOT_FIXED), porque los mensajes sucesivos de la serie se pueden enviar a distintas instancias de la aplicación de servidor.

Si se especifica MQOO_BROWSE o una de las opciones MQOO_INPUT_ * para una cola de clúster, se fuerza al gestor de colas a seleccionar la instancia local de la cola de clúster. Como resultado, el enlace del descriptor de contexto de cola se arregla, incluso si se especifica MQOO_BIND_NOT_FIXED.

Si se especifica MQOO_INQUIRE con MQOO_BIND_NOT_FIXED, las llamadas MQINQ sucesivas que utilizan ese manejador pueden consultar distintas instancias de la cola de clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

MQOO_BIND_NOT_FIXED sólo es válido para las colas y sólo afecta a las colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_ON_GROUP
Permite a una aplicación solicitar que un grupo de mensajes se asigne a la misma instancia de destino.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_AS_Q_DEF
El gestor de colas local enlaza el descriptor de contexto de cola de la forma definida por el atributo de cola **DefBind** . El valor de este atributo es MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED o MQBND_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF es el valor predeterminado cuando no se especifica MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF ayuda a la documentación del programa. No está previsto que esta opción se utilice con cualquiera de las otras dos opciones de vinculación, pero debido a que su valor es cero, no se puede detectar dicho uso.

Opciones de contexto: Las opciones siguientes controlan el proceso del contexto de mensaje:

MQOO_SAVE_ALL_CONTEXT
La información de contexto está asociada con este descriptor de contexto de cola. Esta información se establece a partir del contexto de cualquier mensaje recuperado utilizando este descriptor de contexto. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta información de contexto se puede pasar a un mensaje que luego se coloca en una cola utilizando las llamadas MQPUT o MQPUT1 . Consulte las opciones MQPMO_PASS_IDENTITY_CONTEXT y MQPMO_PASS_ALL_CONTEXT descritas en “MQPMO-Opciones de transferencia de mensajes” en la página 503.

Hasta que un mensaje se haya recuperado correctamente, el contexto no se puede pasar a un mensaje que se está colocando en una cola.

Un mensaje recuperado utilizando una de las opciones de examen MQGMO_BROWSE_ * no tiene guardada su información de contexto (aunque los campos de contexto del parámetro **MsgDesc** se establecen después de un examen).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Debe especificarse una de las opciones MQOO_INPUT_ *.

MQOO_PASS_IDENTITY_CONTEXT
Esto permite especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en el parámetro **PutMsgOpts** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad de una cola de entrada que se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_PASS_ALL_CONTEXT
Esto permite especificar la opción MQPMO_PASS_ALL_CONTEXT en el parámetro **PutMsgOpts** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen de una cola de entrada que se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarlo. Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_SET_IDENTITY_CONTEXT
Esto permite especificar la opción MQPMO_SET_IDENTITY_CONTEXT en el parámetro **PutMsgOpts** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad contenida en el parámetro **MsgDesc** especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarlo. Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_SET_ALL_CONTEXT
Esto permite especificar la opción MQPMO_SET_ALL_CONTEXT en el parámetro **PutMsgOpts** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen contenida en el parámetro **MsgDesc** especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica las siguientes opciones, por lo que no es necesario especificarlas:

- MQOO_PASS_IDENTITY_CONTEXT
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT

Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

Opciones de lectura anticipada:

Cuando se llama a MQOPEN con MQOO_READ_AHEAD, el cliente de IBM MQ sólo permite la lectura anticipada si se cumplen ciertas condiciones. Estas condiciones incluyen:

- El cliente y el gestor de colas remoto deben ser de la IBM WebSphere MQ 7.0 o posterior.
- La aplicación cliente debe compilarse y enlazarse a las bibliotecas de cliente MQI de IBM MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las opciones siguientes controlan si los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Las siguientes notas se aplican a las opciones de lectura anticipada:

- Sólo se puede especificar una de estas opciones.
- Estas opciones sólo son válidas para colas locales, alias y modelo. No son válidos para colas remotas, listas de distribución, temas o gestores de colas.
- Estas opciones sólo son aplicables cuando también se especifica una de las opciones MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE, aunque no es un error especificar estas opciones con MQOO_INQUIRE o MQOO_SET.
- Si la aplicación no se ejecuta como un cliente IBM MQ , estas opciones se ignoran.

MQOO_NO_READ_AHEAD
Los mensajes no persistentes no se envían al cliente antes de que una aplicación los solicite.

MQOO_READ_AHEAD
Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite.

MQOO_READ_AHEAD_AS_Q_DEF
El comportamiento de lectura anticipada viene determinado por el atributo de lectura anticipada predeterminado de la cola que se está abriendo. Éste es el valor predeterminado.

Otras opciones: Las opciones siguientes controlan la comprobación de autorización, lo que sucede cuando el gestor de colas se está desactivando temporalmente, si se debe resolver el nombre de cola local y la multidifusión:

MQOO_ALTERNATE_USER_AUTHORITY
El campo *AlternateUserId* del parámetro **ObjDesc** contiene un identificador de usuario que se debe utilizar para validar esta llamada MQOPEN. La llamada sólo puede realizarse correctamente si este *AlternateUserId* está autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo. Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.

Esta opción es válida para todos los tipos de objeto.

MQOO_FAIL_IF_QUIESCING
La llamada MQOPEN falla si el gestor de colas está en estado de desactivación temporal.

z/OS En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQOPEN falle si la conexión está en estado de desactivación temporal.

Esta opción es válida para todos los tipos de objeto.

Para obtener información sobre los canales de cliente, consulte Visión general de IBM MQ MQI clients.

MQOO_RESOLVE_LOCAL_Q
Rellene ResolvedQName en la estructura MQOD con el nombre de la cola local que se ha abierto. De forma similar, el nombre ResolvedQMgrse rellena con el nombre del gestor de colas local que aloja la cola local. Si la estructura MQOD es inferior a la versión 3, MQOO_RESOLVE_LOCAL_Q se ignora sin que se devuelva ningún error.

La cola local siempre se devuelve cuando se abre una cola local, alias o modelo, pero este no es el caso cuando, por ejemplo, se abre una cola remota o una cola de clúster no local sin la opción MQOO_RESOLVE_LOCAL_Q; el nombre ResolvedQName y ResolvedQMgrse rellenan con el nombre RemoteQName y RemoteQMgrque se encuentra en la definición de cola remota, o de forma similar con la cola de clúster remoto elegida.

Si especifica MQOO_RESOLVE_LOCAL_Q al abrir, por ejemplo, una cola remota, ResolvedQName es la cola de transmisión a la que se colocan los mensajes. El nombre ResolvedQMgrse rellena con el nombre del gestor de colas local que aloja la cola de transmisión.

Si tiene autorización para examinar, entrar o salir en una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQOPEN. No se necesita ninguna autorización especial.

Esta opción sólo es válida para colas y gestores de colas.

MQOO_RESOLVE_LOCAL_TOPIC
Rellene ResolvedQName en la estructura MQOD con el nombre del tema administrativo abierto.

MQOO_NO_MULTIDIFUSIÓN
Los mensajes de publicación no se envían utilizando multidifusión.

Esta opción sólo es válida con la opción MQOO_OUTPUT. Si se especifica sin MQOO_OUTPUT, MQOPEN devuelve con MQRC_OPTIONS_ERROR.

Esta opción sólo es válida para un tema.

Hobj
Tipo: MQHOBJ - salida

Este descriptor de contexto representa el acceso que se ha establecido al objeto. Debe especificarse en las llamadas IBM MQ posteriores que operan en el objeto. Deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto de objeto devuelto es el mismo que el ámbito del descriptor de contexto de conexión especificado en la llamada. Consulte Parámetro MQCONN-Hconn para obtener información sobre el ámbito del descriptor de contexto.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') La cola base de alias no es un tipo válido.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') La comprobación de autorización de la estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_ERROR
(2349, X'92D') La estructura del recurso de acoplamiento no es válida.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2268, X'8DC') Llamadas de colocación inhibidas para todas las colas del clúster.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La cola de transmisión predeterminada no es local.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') Nombre de cola dinámica no válido.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NAME_IN_USE
(2201, X'899 ') Nombre en uso.

MQRC_NAME_NOT_VALID_FOR_TYPE
(2194, X'892 ') El nombre de objeto no es válido para el tipo de objeto.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834 ') El objeto existe.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE
(2042, X'7FA') El objeto ya está abierto con opciones en conflicto.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X' 938 ') Nivel de objeto no compatible.

MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nombre de objeto no válido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Objeto no exclusivo.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nombre de gestor de colas de objeto no válido.

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Los registros de objeto no son válidos.

MQRC_OBJECT_STRING_ERROR
(2441, X'0989 ') El campo Objectstring no es válido

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR
(2044, X'7FC') La estructura de descriptor de objeto no es válida.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_TYPE_ERROR
(2057, X'809 ') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') El nombre de cola remota no es válido.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB, pero se ha especificado una serie de selección que contenía un error de sintaxis.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Cola base alias desconocida.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nombre de objeto desconocido.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestor de colas de objetos desconocido.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestor de colas remoto desconocido.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Cola de transmisión desconocida.

MQRC_ERR_CF_LEVEL
(2366, X'93E') La estructura del recurso de acoplamiento tiene un nivel incorrecto.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Cola de transmisión con un uso incorrecto.

Para obtener información detallada sobre estos códigos, consulte:

- z/OS IBM MQ for z/OS mensajes, códigos de terminación y de razón para IBM MQ for z/OS.
- Mensajes y códigos de razón para todas las demás plataformas IBM MQ excepto z/OS.

Notas de uso general

1. El objeto abierto es uno de los siguientes:
   - Una cola para:
     - Obtener o examinar mensajes (utilizando la llamada MQGET)
     - Transferir mensajes (utilizando la llamada MQPUT)
     - Consultar los atributos de la cola (utilizando la llamada MQINQ)
     - Establecer los atributos de la cola (utilizando la llamada MQSET)

     Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el parámetro **ObjDesc** descrito en “MQOPEN-Abrir objeto” en la página 742.

     Una lista de distribución es un tipo especial de objeto de cola que contiene una lista de colas. Se puede abrir para transferir mensajes, pero no para obtener o examinar mensajes, ni para consultar o establecer atributos. Consulte la nota de uso 8 para obtener más detalles.

Una cola que tiene `QSGDISP(GROUP)` es un tipo especial de definición de cola que no se puede utilizar con las llamadas MQOPEN o MQPUT1 .

- Una lista de nombres para consultar los nombres de las colas de la lista (utilizando la llamada MQINQ).
- Una definición de proceso para consultar sobre los atributos de proceso (utilizando la llamada MQINQ).
- El gestor de colas para consultar los atributos del gestor de colas local (utilizando la llamada MQINQ).
- Un tema para publicar un mensaje (utilizando la llamada MQPUT)

2. Una aplicación puede abrir el mismo objeto más de una vez. Se devuelve un descriptor de objeto diferente para cada apertura. Cada descriptor de contexto que se devuelve se puede utilizar para las funciones para las que se ha realizado la apertura correspondiente.
3. Si el objeto que se está abriendo es una cola distinta de una cola de clúster, toda la resolución de nombres dentro del gestor de colas local tiene lugar en el momento de la llamada MQOPEN. Puede incluir:

   - Resolución del nombre de una definición local de una cola remota al nombre del gestor de colas remoto y el nombre por el que se conoce la cola en el gestor de colas remoto
   - Resolución del nombre del gestor de colas remoto en el nombre de una cola de transmisión local
   - z/OS Sólo en z/OS , la resolución del nombre del gestor de colas remoto al nombre de la cola de transmisión compartida utilizada por el agente de IGQ (sólo se aplica si los gestores de colas local y remoto pertenecen al mismo grupo de compartición de colas)
   - Resolución de alias para el nombre de una cola base o un objeto de tema.

   Sin embargo, tenga en cuenta que las llamadas MQINQ o MQSET posteriores para el descriptor de contexto se relacionan únicamente con el nombre que se ha abierto y no con el objeto resultante después de que se haya producido la resolución de nombres. Por ejemplo, si el objeto abierto es un alias, los atributos devueltos por la llamada MQINQ son los atributos del alias, no los atributos de la cola base o un objeto de tema en el que se resuelve el alias.

   Si el objeto que se está abriendo es una cola de clúster, la resolución de nombres se puede producir en el momento de la llamada MQOPEN, o se puede aplazar hasta más adelante. El punto en el que se produce la resolución está controlado por las opciones MQOO_BIND_ * especificadas en la llamada MQOPEN:

   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NOT_FIXED
   - MQOO_BIND_AS_Q_DEF
   - MQOO_BIND_ON_GROUP

   Consulte Resolución de nombres para obtener más información sobre la resolución de nombres para colas de clúster.
4. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor para examinar, para su uso con llamadas MQGET que especifican el descriptor de objeto y una de las opciones de examinar. Esto permite explorar la cola sin alterar su contenido. Un mensaje que se ha encontrado examinando puede eliminarse de la cola utilizando la opción MQGMO_MSG_UNDER_CURSOR.

   Varios cursores de examen pueden estar activos para una sola aplicación emitiendo varias solicitudes MQOPEN para la misma cola.
5. A las aplicaciones iniciadas por un supervisor desencadenante se les pasa el nombre de la cola asociada con la aplicación cuando se inicia la aplicación. Este nombre de cola se puede especificar en el parámetro **ObjDesc** para abrir la cola. Para conocer detalles, consulte “MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)” en la página 613.

Opciones de lectura anticipada

Cuando se llama a MQOPEN con MQOO_READ_AHEAD, el cliente de IBM MQ sólo permite la lectura anticipada si se cumplen ciertas condiciones. Estas condiciones incluyen:

- El cliente y el gestor de colas remoto deben ser de la IBM WebSphere MQ 7.0 o posterior.
- La aplicación cliente debe compilarse y enlazarse a las bibliotecas de cliente MQI de IBM MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las notas siguientes se aplican al uso de opciones de lectura anticipada.

1. Las opciones de lectura anticipada sólo son aplicables cuando también se especifica una de las opciones MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE. No se genera un error si se especifican opciones de lectura anticipada con las opciones MQOO_ INQUIRE o MQOO_SET.
2. La lectura anticipada no está habilitada cuando se solicita si las opciones utilizadas en la primera llamada MQGET no están soportadas para su uso con la lectura anticipada. Además, la lectura anticipada está inhabilitada cuando el cliente se está conectando a un gestor de colas que no soporta la lectura anticipada.
3. Si la aplicación no se ejecuta como un cliente IBM MQ , las opciones de lectura anticipada se ignoran.

Colas de clúster

Las notas siguientes se aplican al uso de colas de clúster.

1. Cuando una cola de clúster se abre por primera vez, y el gestor de colas local no es un gestor de colas de depósito completo, el gestor de colas local obtiene información sobre la cola de clúster de un gestor de colas de depósito completo. Cuando la red está ocupada, el gestor de colas local puede tardar varios segundos en recibir la información necesaria del gestor de colas de repositorio. Como resultado, es posible que la aplicación que emite la llamada MQOPEN tenga que esperar hasta 10 segundos antes de que se devuelva el control de la llamada MQOPEN. Si el gestor de colas local no recibe la información necesaria sobre la cola de clúster dentro de este tiempo, la llamada falla con el código de razón MQRC_CLUSTER_RESOLUTION_ERROR.
2. Cuando se abre una cola de clúster y hay varias instancias de la cola en el clúster, la instancia abierta depende de las opciones especificadas en la llamada MQOPEN:
   - Si las opciones especificadas incluyen alguna de las siguientes:
     - MQOO_BROWSE
     - MQOO_INPUT_AS_Q_DEF
     - MQOO_INPUT_EXCLUSIVE
     - MQOO_INPUT_SHARED
     - MQOO_SET

     la instancia de la cola de clúster abierta debe ser la instancia local. Si no hay ninguna instancia local de la cola, la llamada MQOPEN falla.
   - Si las opciones especificadas no incluyen ninguna de las opciones descritas anteriormente, pero incluyen una o ambas de las siguientes:
     - MQOO_INQUIRE
     - MQOO_OUTPUT

     la instancia abierta es la instancia local si hay una, y una instancia remota de lo contrario (si se utilizan los valores predeterminados de CLWLUSEQ). Sin embargo, la instancia elegida por el gestor de colas puede ser alterada por una salida de carga de trabajo de clúster (si la hay).

3. Si hay una suscripción para la cola, pero no la reconoce un repositorio completo, el objeto no está presente en el clúster y la llamada falla con el código de razón MQRC_OBJECT_NAME.

Para obtener más información sobre las colas de clúster, consulte Colas de clúster.

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución están soportadas en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

y para IBM MQ MQI clients conectados a estos sistemas.

1. Los campos de la estructura MQOD deben establecerse de la forma siguiente al abrir una lista de distribución:
   - `Version` debe ser MQOD_VERSION_2 o superior.
   - `ObjectType` debe ser MQOT_Q.
   - `ObjectName` debe estar en blanco o la serie nula.
   - `ObjectQMgrName` debe estar en blanco o la serie nula.
   - `RecsPresent` Tiene que ser mayor que cero.
   - Uno de `ObjectRecOffset` y `ObjectRecPtr` debe ser cero y el otro distinto de cero.
   - No más de uno de `ResponseRecOffset` y `ResponseRecPtr` puede ser distinto de cero.
   - Debe haber `RecsPresent` registros de objeto, direccionado por `ObjectRecOffset` o `ObjectRecPtr`. Los registros de objeto deben establecerse en los nombres de las colas de destino que se van a abrir.
   - Si uno de `ResponseRecOffset` y `ResponseRecPtr` es distinto de cero, debe haber `RecsPresent` registros de respuesta presentes. Los establece el gestor de colas si la llamada se completa con el código de razón MQRC_MULTIPLE_REASON.

   Un MQOD version-2 también se puede utilizar para abrir una única cola que no esté en una lista de distribución, asegurándose de que `RecsPresent` sea cero.
2. Sólo las siguientes opciones de apertura son válidas en el parámetro **`Options`** :
   - MQOO_OUTPUT
   - MQOO_PASS_ * _CONTEXTO
   - MQOO_SET_ * _CONTEXTO
   - MQOO_ALTERNATE_USER_AUTHORITY
   - MQOO_FAIL_IF_QUIESCING
3. Las colas de destino de la lista de distribución pueden ser colas locales, alias o remotas, pero no pueden ser colas modelo. Si se especifica una cola modelo, dicha cola no se puede abrir, con el código de razón MQRC_Q_TYPE_ERROR. Sin embargo, esto no impide que otras colas de la lista se abran correctamente.
4. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:
   - Si las operaciones abiertas para las colas de la lista de distribución son satisfactorias o fallan de la misma forma, los parámetros de código de terminación y código de razón se establecen para

describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si cada apertura es satisfactoria, el código de terminación se establece en MQCC_OK y el código de razón se establece en MQRC_NONE; si cada apertura falla porque no existe ninguna de las colas, los parámetros se establecen en MQCC_FAILED y MQRC_UNKNOWN_OBJECT_NAME.

- Si las operaciones abiertas para las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
  - El parámetro de código de finalización se establece en MQCC_WARNING si al menos una apertura se ha realizado correctamente y en MQCC_FAILED si todo ha fallado.
  - El parámetro de código de razón se establece en MQRC_MULTIPLE_REASON.
  - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

5. Cuando una lista de distribución se ha abierto correctamente, el descriptor de contexto `Hobj` devuelto por la llamada se puede utilizar en llamadas MQPUT posteriores para colocar mensajes en las colas de la lista de distribución y en una llamada MQCLOSE para renunciar al acceso a la lista de distribución. La única opción de cierre válida para una lista de distribución es MQCO_NONE.

   La llamada MQPUT1 también se puede utilizar para colocar un mensaje en una lista de distribución; la estructura MQOD que define las colas de la lista se especifica como un parámetro en dicha llamada.
6. Cada destino abierto correctamente en la lista de distribución cuenta como un descriptor de contexto independiente al comprobar si la aplicación ha superado el número máximo permitido de descriptores de contexto (consulte el atributo de gestor de colas **`MaxHandles`**). Esto es cierto incluso cuando dos o más de los destinos de la lista de distribución se resuelven en la misma cola física. Si la llamada MQOPEN o MQPUT1 para una lista de distribución haría que el número de descriptores de contexto utilizados por la aplicación excediera `MaxHandles`, la llamada falla con el código de razón MQRC_HANDLE_NOT_AVAILABLE.
7. Cada destino que se abre correctamente tiene el valor de su atributo **`OpenOutputCount`** incrementado en uno. Si dos o más de los destinos de la lista de distribución se resuelven en la misma cola física, el atributo **`OpenOutputCount`** de dicha cola se incrementa en el número de destinos de la lista de distribución que se resuelven en dicha cola.
8. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.
9. Una lista de distribución sólo puede contener un destino.

Colas remotas

Las notas siguientes se aplican al uso de colas remotas.

Una cola remota se puede especificar de una de dos maneras en el parámetro **`ObjDesc`** de esta llamada.

- Especificando para `ObjectName` el nombre de una definición local de la cola remota. En este caso, `ObjectQMgrName` hace referencia al gestor de colas local y se puede especificar como espacios en blanco o (en el lenguaje de programación C) como una serie nula.

  La validación de seguridad realizada por el gestor de colas local verifica que el usuario tiene autorización para abrir la definición local de la cola remota.
- Especificando para `ObjectName` el nombre de la cola remota tal como la conoce el gestor de colas remoto. En este caso, `ObjectQMgrName` es el nombre del gestor de colas remoto.

  La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado a enviar mensajes a la cola de transmisión como resultado del proceso de resolución de nombres.

En cualquier caso:

- El gestor de colas local no envía mensajes al gestor de colas remoto para comprobar que el usuario tiene autorización para colocar mensajes en la cola.
- Cuando llega un mensaje al gestor de colas remoto, el gestor de colas remoto puede rechazarlo porque el usuario que ha originado el mensaje no está autorizado.

Consulte los campos `ObjectName` y `ObjectQMgrName` descritos en "MQOD-Descriptor de objeto" en la página 483 para obtener más información.

Objetos

Seguridad

Las notas siguientes están relacionadas con los aspectos de seguridad del uso de MQOPEN.

El gestor de colas realiza comprobaciones de seguridad cuando se emite una llamada MQOPEN, para verificar que el identificador de usuario bajo el que se ejecuta la aplicación tiene el nivel de autorización adecuado antes de que se permita el acceso. La comprobación de autorización se realiza en el nombre del objeto que se está abriendo, y no en el nombre o nombres, lo que resulta después de que se haya resuelto un nombre.

Si el objeto que se está abriendo es una cola alias que apunta a un objeto de tema, el gestor de colas realiza una comprobación de seguridad en el nombre de cola alias, antes de realizar una comprobación de seguridad para el tema como si el objeto de tema se hubiera utilizado directamente.

Si el objeto que se está abriendo es un objeto de tema, ya sea con `ObjectName` solo o utilizando `ObjectString` (con o sin una base `ObjectName`), el gestor de colas realiza la comprobación de seguridad utilizando la serie de tema resultante, tomada del objeto de tema especificado en `ObjectName`y, si es necesario, concatenándola con la proporcionada en `ObjectString`y, a continuación, buscando el objeto de tema más cercano en o por encima de ese punto del árbol de temas para realizar la comprobación de seguridad. Es posible que no sea el mismo objeto de tema que se ha especificado en `ObjectName`.

Si el objeto que se está abriendo es una cola modelo, el gestor de colas realiza una comprobación de seguridad completa con el nombre de la cola modelo y el nombre de la cola dinámica que se crea. Si la cola dinámica resultante se abre entonces de forma explícita, se realiza una comprobación de seguridad de recursos adicional con respecto al nombre de la cola dinámica.

z/OS En z/OS, el gestor de colas sólo realiza comprobaciones de seguridad si la seguridad está habilitada. Para obtener más información sobre la comprobación de seguridad, consulte Configuración de la seguridad en z/OS .

Atributos

Las notas siguientes están relacionadas con los atributos.

Los atributos de un objeto pueden cambiar mientras una aplicación tiene el objeto abierto. En muchos casos, la aplicación no se da cuenta de ello, pero para determinados atributos el gestor de colas marca el descriptor de contexto como ya no válido. Estos atributos son:

- Cualquier atributo que afecte a la resolución de nombres del objeto. Esto se aplica independientemente de las opciones de apertura utilizadas e incluye lo siguiente:
  - Un cambio en el atributo **BaseQName** de una cola alias que está abierta.
  - Un cambio en el atributo **TargetType** de una cola alias que está abierta.
  - Un cambio en los atributos de cola **RemoteQName** o **RemoteQMgrName** , para cualquier descriptor de contexto que esté abierto para esta cola, o para una cola que se resuelva a través de esta definición como un alias de gestor de colas.
  - Cualquier cambio que haga que un descriptor de contexto abierto actualmente para una cola remota se resuelva en una cola de transmisión diferente, o que no se pueda resolver en una en absoluto. Por ejemplo, esto puede incluir:

- Un cambio en el atributo **XmitQName** de la definición local de una cola remota, si la definición se está utilizando para una cola o para un alias de gestor de colas.
- z/OS Sólo en z/OS , un cambio en el valor del atributo del gestor de colas **IntraGroupqueuing** , o un cambio en la definición de la cola de transmisión compartida (SYSTEM.QSG.TRANSMIT.QUEUE) utilizado por el agente de IGQ.

Hay una excepción a esto: la creación de una nueva cola de transmisión. Un descriptor de contexto que se habría resuelto en esta cola si estuviera presente cuando se abrió el descriptor de contexto, pero que en su lugar se hubiera resuelto en la cola de transmisión predeterminada, no se ha convertido en no válido.

– Un cambio en el atributo del gestor de colas **DefXmitQName** . En este caso, todos los descriptores de contexto abiertos que se han resuelto en la cola especificada anteriormente (que se ha resuelto sólo porque era la cola de transmisión predeterminada) se marcan como no válidos. Los manejadores que se han resuelto en esta cola por otras razones no se ven afectados.

- El atributo de cola **Shareability** , si hay dos o más descriptores de contexto que proporcionan actualmente acceso MQOO_INPUT_SHARED para esta cola, o para una cola que se resuelve en esta cola. Si es así, todos los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, se marcan como no válidos, independientemente de las opciones de apertura.

  z/OS En z/OS, los descriptores de contexto descritos anteriormente se marcan como no válidos si uno o varios descriptores de contexto proporcionan actualmente acceso MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE a la cola.
- El atributo de cola **Usage** , para todos los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, independientemente de las opciones de apertura.

Cuando un descriptor de contexto se marca como no válido, todas las llamadas posteriores (distintas de MQCLOSE) que utilizan este descriptor de contexto fallan con el código de razón MQRC_OBJECT_CHANGED. La aplicación debe emitir una llamada MQCLOSE (utilizando el descriptor de contexto original) y, a continuación, volver a abrir la cola. Las actualizaciones no confirmadas en el descriptor de contexto antiguo de las llamadas satisfactorias anteriores se pueden seguir confirmando o restituyendo, según lo requiera la lógica de la aplicación.

Si el cambio de un atributo hace que esto suceda, utilice una versión de fuerza especial de la llamada.

Invocación en C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
       &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQOD     ObjDesc;   /* Object descriptor */
MQLONG   Options;   /* Options that control the action of MQOPEN */
MQHOBJ   Hobj;      /* Object handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Object descriptor
```

```
 01  OBJDESC.
     COPY CMQODV.
**   Options that control the action of MQOPEN
 01  OPTIONS   PIC S9(9) BINARY.
**   Object handle
 01  HOBJ      PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl ObjDesc   like MQOD;      /* Object descriptor */
dcl Options   fixed bin(31);  /* Options that control the action of
                                 MQOPEN */
dcl Hobj      fixed bin(31);  /* Object handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS      F  Connection handle
OBJDESC   CMQODA  ,  Object descriptor
OPTIONS   DS      F  Options that control the action of MQOPEN
HOBJ      DS      F  Object handle
COMPCODE  DS      F  Completion code
REASON    DS      F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

Windows

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim ObjDesc  As MQOD 'Object descriptor'
Dim Options  As Long 'Options that control the action of MQOPEN'
Dim Hobj     As Long 'Object handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQPUT-Colocar mensaje

La llamada MQPUT coloca un mensaje en una cola o lista de distribución, o en un tema. La cola, la lista de distribución o el tema ya deben estar abiertos.

Sintaxis

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de `Hconn` ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

z/OS En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este descriptor de contexto representa la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. El valor de `Hobj` lo ha devuelto una llamada MQOPEN anterior que ha especificado la opción MQOO_OUTPUT.

MsgDesc

Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información sobre el mensaje una vez completada la solicitud de colocación. Consulte "MQMD - Descriptor de mensaje" en la página 421 para obtener detalles.

Si la aplicación proporciona un MQMD version-1 , los datos de mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *Formato* de MQMD debe establecerse en MQFMT_MD_EXTENSION para indicar que existe un MQMDE. Consulte "MQMDE-Extensión de descriptor de mensaje" en la página 474 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se proporciona un descriptor de mensaje válido en los campos `OriginalMsgHandle` o `NewMsgHandle` de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los manejadores de mensajes.

Si utiliza o tiene previsto utilizar salidas de API, le recomendamos que proporcione explícitamente una estructura MQMD y no utilice los descriptores de mensaje asociados con los descriptores de mensaje. Esto se debe a que la salida de API asociada a la llamada MQPUT o MQPUT1 no puede determinar qué valores MQMD utiliza el gestor de colas para completar la solicitud MQPUT o MQPUT1 .

PutMsg

Tipo: MQPMO-entrada/salida

Consulte "MQPMO-Opciones de transferencia de mensajes" en la página 503 para obtener los detalles.

BufferLength

Tipo: MQLONG - entrada

La longitud del mensaje en `Buffer`. Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior para `BufferLength` depende de varios factores:

- Si el destino es una cola local o se resuelve en una cola local, el límite superior depende de si:
  - El gestor de colas local da soporte a la segmentación.

– La aplicación emisora especifica el distintivo que permite al gestor de colas segmentar el mensaje. Este distintivo es MQMF_SEGMENTATION_ALLOWED y se puede especificar en un MQMD de version-2 o en un MQMDE utilizado con un MQMD de version-1 .

Si se cumplen ambas condiciones, `BufferLength` no puede exceder de 999 999 999 menos el valor del campo `Offset` en MQMD. El mensaje lógico más largo que se puede transferir es, por lo tanto, 999 999 999 bytes (cuando `Offset` es cero). Sin embargo, las restricciones de recursos impuestas por el sistema operativo o el entorno en el que se ejecuta la aplicación pueden dar como resultado un límite inferior.

Si no se cumple una o ambas de las condiciones anteriores, `BufferLength` no puede exceder el atributo **MaxMsgLength** y el atributo **MaxMsgLength** del gestor de colas.

- Si el destino es una cola remota o se resuelve en una cola remota, se aplican las condiciones para las colas locales, pero en cada gestor de colas a través del cual debe pasar el mensaje para llegar a la cola de destino; en particular:

  1. La cola de transmisión local utilizada para almacenar el mensaje temporalmente en el gestor de colas local
  2. Colas de transmisión intermedias (si las hay) utilizadas para almacenar el mensaje en los gestores de colas de la ruta entre los gestores de colas local y de destino
  3. La cola de destino en el gestor de colas de destino

  Por lo tanto, el mensaje más largo que se puede transferir está gobernado por el más restrictivo de estas colas y gestores de colas.

  Cuando un mensaje está en una cola de transmisión, la información adicional reside en los datos del mensaje, y esto reduce la cantidad de datos de aplicación que pueden transportarse. En esta situación, reste los bytes MQ_MSG_HEADER_LENGTH de los valores `MaxMsgLength` de las colas de transmisión al determinar el límite para `BufferLength`.

  Nota: Sólo se puede diagnosticar de forma síncrona el incumplimiento de la condición 1 (con el código de razón MQRC_MSG_TOO_BIG_FOR_Q o MQRC_MSG_TOO_BIG_FOR_Q_MGR) cuando se transfiere el mensaje. Si no se cumplen las condiciones 2 o 3, el mensaje se redirige a una cola de mensajes no entregados, ya sea en un gestor de colas intermedio o en el gestor de colas de destino. Si esto sucede, se genera un mensaje de informe si el remitente lo ha solicitado.

Almacenamiento intermedio
Tipo: MQBYTExBufferLongitud-entrada

Se trata de un almacenamiento intermedio que contiene los datos de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si `Buffer` contiene datos numéricos o de caracteres, establezca los campos `CodedCharSetId` y `Encoding` del parámetro **MsgDesc** en los valores adecuados para los datos; esto permite al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT deben estar en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE).

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BufferLength** es cero, no se hace referencia a `Buffer` ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica CompCode.

Si CompCode es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si CompCode es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Especificación de persistencia incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838 ') No se reconocen las opciones de informe en el descriptor de mensaje.

Si CompCode es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_TARGTYPE_CAMBIADO
(2480, X'09B0') El tipo de destino de suscripción ha cambiado de cola a objeto de tema.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT o MQCMIT se ha interrumpido y el proceso de reconexión no puede restablecer un resultado definido.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR
(2416, X' 970 ') La estructura de parámetro de grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR
(2235, X'8BB') La estructura de cabecera PCF no es válida.

MQRC_CFIF_ERROR
(2414, X'96E') La estructura del parámetro de filtro de enteros PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR
(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR
(2237, X'8BD') Estructura de parámetro de entero PCF o estructura de parámetro de entero PCIF*64 no válida.

MQRC_CFSF_ERROR
(2415, X'96F') La estructura del parámetro de filtro de serie PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR
(2238, X'8BE') La estructura de parámetro de lista de series PCF no es válida.

MQRC_CFST_ERROR
(2239, X'8BF') La estructura del parámetro de serie PCF no es válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

ERROR DE MQRC_CONTENT_
2554 (X'09FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje debe entregarse a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DH_ERROR
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

MQRC_DLH_ERROR
(2141, X'85D') La estructura de cabecera de letra muerda no es válida.

MQRC_EPH_ERROR
(2420, X' 974 ') La estructura PCF incorporada no es válida.

MQRC_EXPIRY_ERROR
(2013, X'7DD') Tiempo de caducidad no válido.

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Código de comentarios no válido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HEADER_ERROR
(2142, X'85E') La estructura de cabecera MQ no es válida.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_IIH_ERROR
(2148, X'864 ') IMS estructura de cabecera de información no válida.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Especificación de persistencia incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Falta la cola de respuesta o se ha utilizado MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH
(2332, X'91C') Los datos de mensaje no empiezan por MQWIH.

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Los distintivos de mensaje no son válidos.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Cola no abierta para salida.

MQRC_NOT_OPEN_FOR_PASS_ALL
(2093, X'82D') Cola no abierta para pasar todo el contexto.

MQRC_NOT_OPEN_FOR_PASS_IDENT
(2094, X'82E') La cola no está abierta para el contexto de identidad de paso.

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') Cola no abierta para establecer todo el contexto.

MQRC_NOT_OPEN_FOR_SET_IDENT
(2096, X'830 ') La cola no está abierta para el contexto de identidad establecido.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

ERROR DE MQRC_OFFSET_
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR
(2149, X'865 ') Estructuras PCF no válidas.

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

MQRC_PRIORITY_ERROR
(2050, X'802 ') La prioridad del mensaje no es válida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola, para la cola en la que se resuelve esta cola o el tema.

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

MQRC_PUT_NO_RETENIDO
(2479, X'09AF') No se ha podido retener la publicación

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Después de la reconexión, se ha producido un error al restablecer los descriptores de contexto para una conexión reconectable.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_RFH_ERROR
(2334, X'91E') La estructura MQRFH o MQRFH2 no es válida.

MQRC_RMH_ERROR
(2220, X'8AC') La estructura de cabecera de mensaje de referencia no es válida.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

MQRC_SEGMENTS_NOT_SUPPORTED
(2365, X'93D') Segmentos no soportados.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si se envía la publicación al suscriptor.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

ERROR MQRC_TM
(2265, X'8D9') La estructura de mensajes del desencadenante no es válida.

ERROR MQRC_TMC
(2191, X'88F') La estructura del mensaje desencadenante de caracteres no es válida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR
(2333, X'91D') La estructura MQWIH no es válida.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XQH_ERROR
(2260, X'8D4') La estructura de cabecera de cola de transmisión no es válida.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso de temas

1. Las notas siguientes se aplican al uso de temas:

   a. Cuando se utiliza MQPUT para publicar mensajes sobre un tema, donde uno o más suscriptores de ese tema no pueden recibir la publicación debido a un problema con su cola de suscriptores (por ejemplo, está llena), el código de razón devuelto a la llamada MQPUT y el comportamiento de entrega depende del valor de los atributos PMSGDLV o NPMSGDLV en el TOPIC. Tenga en cuenta que la entrega de una publicación a la cola de mensajes no entregados cuando se especifica MQRO_DEAD_LETTER_Q, o descartar el mensaje cuando se especifica MQRO_DISCARD_MSG, se considera una entrega correcta del mensaje. Si no se entrega ninguna de las publicaciones, MQPUT devuelve MQRC_PUBLICATION_FAILURE. Esto puede ocurrir en los siguientes casos:

      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALL y cualquier suscripción (duradera o no) tiene una cola que no puede recibir la publicación.
      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción duradera tiene una cola que no puede recibir la publicación.

MQPUT puede devolver con MQRC_NONE aunque las publicaciones no se hayan podido entregar a algunos suscriptores en los casos siguientes:

- Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLAVAIL y cualquier suscripción, duradera o no, tiene una cola que no puede recibir la publicación.
- Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (en función de la persistencia del mensaje) establecido en ALLDUR y una suscripción no duradera tiene una cola que no puede recibir la publicación.

Puede utilizar el atributo de tema USEDLQ para determinar si se utiliza la cola de mensajes no entregados cuando los mensajes de publicación no se pueden entregar a su cola de suscriptor correcta. Para obtener más información sobre el uso de USEDLQ, consulte DEFINE TOPIC.

b. Si no hay suscriptores para el tema que se está utilizando, el mensaje publicado no se envía a ninguna cola y se descarta. No importa si el mensaje es persistente o no persistente, o si tiene una caducidad ilimitada o tiene una hora de caducidad, todavía se descarta si no hay suscriptores. La excepción a esto es si el mensaje se va a retener, en cuyo caso, aunque no se envía a las colas de ningún suscriptor, se almacena en el tema que se va a entregar a cualquier suscripción nueva o a cualquier suscriptor que solicite publicaciones retenidas utilizando MQSUBRQ.

MQPUT y MQPUT1

Puede utilizar las llamadas MQPUT y MQPUT1 para colocar mensajes en una cola; la llamada a utilizar depende de las circunstancias

- Utilice la llamada MQPUT para colocar varios mensajes en la misma cola.

  En primer lugar, se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .
- Utilice la llamada MQPUT1 para poner sólo un mensaje en una cola.

  Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

Colas de destino

Las notas siguientes se aplican al uso de colas de destino:

1. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de esos mensajes se conserva si se cumplen las condiciones detalladas. Algunas condiciones se aplican tanto a las colas de destino locales como a las remotas; otras condiciones se aplican sólo a las colas de destino remotas.

   Condiciones que se aplican a las colas de destino locales y remotas

   - Todas las llamadas MQPUT están dentro de la misma unidad de trabajo, o ninguna de ellas está dentro de una unidad de trabajo.

     Tenga en cuenta que cuando los mensajes se colocan en una cola determinada dentro de una sola unidad de trabajo, los mensajes de otras aplicaciones pueden intercalarse con la secuencia de mensajes de la cola.
   - Todas las llamadas MQPUT se realizan utilizando el mismo descriptor de objeto *Hobj*.

     En algunos entornos, la secuencia de mensajes también se conserva cuando se utilizan distintos manejadores de objetos, si las llamadas se realizan desde la misma aplicación. El significado de *la misma aplicación* viene determinado por el entorno:

     – z/OS En z/OS, la aplicación es:

       - Para CICS, la tarea CICS
       - Para IMS, la tarea

- Para el lote z/OS , la tarea
  - IBM i En IBM i, la aplicación es el trabajo.
  - Windows UNIX En Windows y UNIX, la aplicación es la hebra.
- Los mensajes tienen todos la misma prioridad.
- Los mensajes no se colocan en una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el atributo de cola DefBind tiene el valor MQBND_BIND_NOT_FIXED).

Condiciones adicionales que se aplican a las colas de destino remoto

- Sólo hay una vía de acceso desde el gestor de colas emisor al gestor de colas de destino.

  Si algunos mensajes de la secuencia pueden ir a una vía de acceso diferente (por ejemplo, debido a la reconfiguración, el equilibrio de tráfico o la selección de vía de acceso en función del tamaño del mensaje), el orden de los mensajes en el gestor de colas de destino no se puede garantizar.
- Los mensajes no se colocan temporalmente en colas de mensajes no entregados en los gestores de colas de envío, intermedios o de destino.

  Si uno o varios de los mensajes se colocan temporalmente en una cola de mensajes no entregados (por ejemplo, porque una cola de transmisión o la cola de destino está llena temporalmente), los mensajes pueden llegar a la cola de destino fuera de secuencia.
- Los mensajes son todos persistentes o todos no persistentes.

  Si un canal de la ruta entre los gestores de colas de envío y de destino tiene su atributo **NonPersistentMsgSpeed** establecido en MQNPMS_FAST, los mensajes no persistentes pueden saltar por delante de los mensajes persistentes, lo que da como resultado que el orden de los mensajes persistentes relativos a los mensajes no persistentes no se conserve. Sin embargo, se conserva el orden de los mensajes persistentes relativos entre sí y de los mensajes no persistentes relativos entre sí.

Si estas condiciones no se cumplen, puede utilizar grupos de mensajes para conservar el orden de los mensajes, pero esto requiere que las aplicaciones de envío y recepción utilicen el soporte de agrupación de mensajes. Para obtener más información sobre los grupos de mensajes, consulte:

- MQMD - Campo MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución están soportadas en los entornos siguientes:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Solaris Solaris
- Windows Windows

y para IBM MQ MQI clients conectados a estos sistemas.

1. Puede colocar mensajes en una lista de distribución utilizando version-1 o un MQPMO version-2 . Si utiliza un MQPMO version-1 (o un MQPMO version-2 con `RecsPresent` igual a cero), la aplicación no puede proporcionar registros de mensajes de colocación ni registros de respuesta. No puede

identificar las colas que encuentran errores si el mensaje se envía correctamente a algunas colas de la lista de distribución y no a otras.

Si la aplicación proporciona registros de mensajes de colocación o registros de respuesta, establezca el campo `Version` en MQPMO_VERSION_2.

También puede utilizar un MQPMO version-2 para enviar mensajes a una sola cola que no esté en una lista de distribución, asegurándose de que `RecsPresent` sea cero.

2. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:
   - Si todas las colocaciones en las colas de la lista de distribución son satisfactorias o fallan de la misma forma, el código de terminación y los parámetros de código de razón se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

     Por ejemplo, si cada transferencia es satisfactoria, el código de terminación y el código de razón se establecen en MQCC_OK y MQRC_NONE; si cada transferencia falla porque todas las colas están inhibidas para la transferencia, los parámetros se establecen en MQCC_FAILED y MQRC_PUT_inhibitITED.
   - Si las colocaciones en las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
     - El parámetro de código de terminación se establece en MQCC_WARNING si al menos una operación de transferencia ha sido satisfactoria, y en MQCC_FAILED si todas las operaciones han fallado.
     - El parámetro de código de razón se establece en MQRC_MULTIPLE_REASON.
     - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

     Si la colocación en un destino falla porque la apertura para dicho destino ha fallado, los campos del registro de respuesta se establecen en MQCC_FAILED y MQRC_OPEN_FAILED; dicho destino se incluye en `InvalidDestCount`.
3. Si un destino de la lista de distribución se resuelve en una cola local, el mensaje se coloca en esa cola en formato normal (es decir, no como un mensaje de lista de distribución). Si más de un destino se resuelve en la misma cola local, se coloca un mensaje en la cola para cada destino.

   Si un destino de la lista de distribución se resuelve en una cola remota, se coloca un mensaje en la cola de transmisión adecuada. Cuando varios destinos se resuelven en la misma cola de transmisión, se puede colocar en la cola de transmisión un único mensaje de lista de distribución que contenga esos destinos, incluso si esos destinos no estaban adyacentes en la lista de destinos proporcionada por la aplicación. Sin embargo, esto sólo se puede hacer si la cola de transmisión da soporte a mensajes de lista de distribución (consulte DistLists).

   Si la cola de transmisión no soporta listas de distribución, se coloca una copia del mensaje en formato normal en la cola de transmisión para cada destino que utiliza dicha cola de transmisión.

   Si una lista de distribución con los datos de mensaje de aplicación es demasiado grande para una cola de transmisión, el mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños, cada uno de los cuales contiene menos destinos. Si los datos del mensaje de aplicación solo se ajustan a la cola, los mensajes de lista de distribución no se pueden utilizar en absoluto, y el gestor de colas genera una copia del mensaje en formato normal para cada destino que utiliza dicha cola de transmisión.

   Si distintos destinos tienen una prioridad de mensaje o una persistencia de mensaje diferentes (esto puede ocurrir cuando la aplicación especifica MQPRI_PRIORITY_AS_Q_DEF o MQPER_PERSISTENCE_AS_Q_DEF), los mensajes no se conservan en el mismo mensaje de lista de distribución. En su lugar, el gestor de colas genera tantos mensajes de lista de distribución como sean necesarios para acomodar los diferentes valores de prioridad y persistencia.
4. Una colocación en una lista de distribución puede dar como resultado:
   - Un único mensaje de lista de distribución, o

- Un número de mensajes de lista de distribución más pequeños, o
- Una combinación de mensajes de lista de distribución y mensajes normales, o
- Sólo mensajes normales.

Cuál de los anteriores se produce depende de si:

- Los destinos de la lista son locales, remotos o una mezcla.
- Los destinos tienen la misma prioridad de mensaje y persistencia de mensaje.
- Las colas de transmisión pueden contener mensajes de lista de distribución.
- Las longitudes máximas de mensajes de las colas de transmisión son lo suficientemente grandes para acomodar el mensaje en formato de lista de distribución.

Sin embargo, independientemente de cuál de las situaciones anteriores, cada mensaje *físico* resultante (es decir, cada mensaje normal o mensaje de lista de distribución resultante de la operación de transferir) cuenta como un solo mensaje *uno* cuando:

- Comprobación de si la aplicación ha superado el número máximo permitido de mensajes en una unidad de trabajo (consulte el atributo del gestor de colas **MaxUncommittedMsgs**).
- Comprobando si se cumplen las condiciones de desencadenamiento.
- Incrementando las profundidades de cola y comprobando si se superaría la profundidad máxima de cola de las colas.

5. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.

Cabeceras

Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Si el gestor de colas detecta un error, la llamada falla con un código de razón adecuado. Las comprobaciones realizadas varían en función de las estructuras concretas que estén presentes:

- Las comprobaciones sólo se realizan si se utiliza un MQMD version-2 o posterior en la llamada MQPUT o MQPUT1 . Las comprobaciones no se realizan si se utiliza un MQMD version-1 , aunque haya un MQMDE al principio de los datos del mensaje.
- Las estructuras que no están soportadas por el gestor de colas local y las estructuras que siguen a la primera MQDLH del mensaje no se validan.
- El gestor de colas valida completamente las estructuras MQDH y MQMDE.
- El gestor de colas valida parcialmente otras estructuras (no se comprueban todos los campos).

Las comprobaciones generales realizadas por el gestor de colas son las siguientes:

- El campo `StrucId` debe ser válido.
- El campo `Version` debe ser válido.
- El campo `StrucLength` debe especificar un valor lo suficientemente grande para incluir la estructura más cualquier dato de longitud variable que forme parte de la estructura.
- El campo `CodedCharSetId` no debe ser cero, o un valor negativo que no es válido (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR y MQCCSI_UNDEFINED no son válidos en la mayoría de las estructuras de cabecera de IBM MQ).
- El parámetro **BufferLength** de la llamada debe especificar un valor que sea lo suficientemente grande para incluir la estructura (la estructura no debe extenderse más allá del final del mensaje).

Además de los controles generales de las estructuras, deberán cumplirse las siguientes condiciones:

- La suma de las longitudes de las estructuras de un mensaje PCF debe ser igual a la longitud especificada por el parámetro **BufferLength** en la llamada MQPUT o MQPUT1 . Un mensaje PCF es un mensaje que tiene un nombre de formato de MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.
- Una estructura IBM MQ no se debe truncar, excepto en las situaciones siguientes en las que se permiten estructuras truncadas:
  - Mensajes que son mensajes de informe.
  - Mensajes PCF.
  - Mensajes que contienen una estructura MQDLH. (Las estructuras que siguen a la primera MQDLH se pueden truncar; las estructuras que preceden a la MQDLH no.)
- Una estructura IBM MQ no debe dividirse en dos o más segmentos; la estructura debe estar contenida por completo en un segmento.

Almacenamiento intermedio

Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

- Si el tamaño del parámetro **Buffer** es menor que la longitud especificada por el parámetro **BufferLength** , la llamada falla con el código de razón MQRC_BUFFER_LENGTH_ERROR.
- El parámetro **Buffer** se declara como de tipo `String`. Si los datos que se van a colocar en la cola no son del tipo `String`, utilice elLlamada MQPUTAny en lugar de MQPUT.

  La llamada MQPUTAny tiene los mismos parámetros que la llamada MQPUT, excepto que el parámetro **Buffer** se declara como de tipo Any, lo que permite colocar cualquier tipo de datos en la cola. Sin embargo, esto significa que `Buffer` no se puede comprobar para asegurarse de que tiene como mínimo `BufferLength` bytes de tamaño.

Invocación en C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQHOBJ   Hobj;          /* Object handle */
MQMD     MsgDesc;       /* Message descriptor */
MQPMO    PutMsgOpts;    /* Options that control the action of MQPUT */
MQLONG   BufferLength;  /* Length of the message in Buffer */
MQBYTE   Buffer[n];     /* Message data */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                        BUFFER, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object handle
 01  HOBJ          PIC S9(9) BINARY.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQPUT
 01  PUTMSGOPTS.
     COPY CMQPMOV.
```

```
**   Length of the message in BUFFER
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Message data
 01  BUFFER        PIC X(n).
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl Hobj          fixed bin(31);  /* Object handle */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl PutMsgOpts    like MQPMO;     /* Options that control the action of
                                     MQPUT */
dcl BufferLength  fixed bin(31);  /* Length of the message in Buffer */
dcl Buffer        char(n);        /* Message data */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQPUT,(HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
               BUFFER,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN         DS       F      Connection handle
HOBJ          DS       F      Object handle
MSGDESC       CMQMDA   ,      Message descriptor
PUTMSGOPTS    CMQPMOA  ,      Options that control the action of MQPUT
BUFFERLENGTH  DS       F      Length of the message in BUFFER
BUFFER        DS       CL(n)  Message data
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Invocación en Visual Basic

Windows

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn        As Long   'Connection handle'
Dim Hobj         As Long   'Object handle'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim PutMsgOpts   As MQPMO  'Options that control the action of MQPUT'
Dim BufferLength As Long   'Length of the message in Buffer'
Dim Buffer       As String 'Message data'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQPUT1 -Colocar un mensaje

La llamada MQPUT1 coloca un mensaje en una cola, o lista de distribución, o en un tema.

No es necesario que la cola, la lista de distribución o el tema estén abiertos.

Sintaxis

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

z/OS En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

ObjDesc
Tipo: MQOD-entrada/salida

Es una estructura que identifica la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. Consulte "MQOD-Descriptor de objeto" en la página 483 para obtener detalles.

Si la estructura es una cola, el usuario debe tener autorización para abrir la cola para salida. La cola no debe ser una cola modelo.

MsgDesc
Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información de retorno una vez completada la solicitud de colocación. Consulte "MQMD - Descriptor de mensaje" en la página 421 para obtener detalles.

Si la aplicación proporciona un MQMD version-1 , los datos de mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. Establezca el campo `Format` de MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Consulte "MQMDE-Extensión de descriptor de mensaje" en la página 474 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se proporciona un manejador de mensajes válido en el campo `MsgHandle` de la estructura MQGMO o en los campos `OriginalMsgHandle` o `NewMsgHandle` de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los manejadores de mensajes.

PutMsg
Tipo: MQPMO-entrada/salida

Consulte "MQPMO-Opciones de transferencia de mensajes" en la página 503 para obtener los detalles.

BufferLength
Tipo: MQLONG - entrada

La longitud del mensaje en `Buffer`. Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior depende de diversos factores; consulte "MQPUT-Colocar mensaje" en la página 760 para obtener la descripción del parámetro **`BufferLength`** .

Almacenamiento intermedio
Tipo: MQBYTExBufferLongitud-entrada

Es un almacenamiento intermedio que contiene los datos de mensaje de aplicación que se van a enviar. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si `Buffer` contiene datos numéricos o de caracteres, establezca los campos `CodedCharSetId` y `Encoding` del parámetro **MsgDesc** en los valores adecuados para los datos; esto permite al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT1 deben estar en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE).

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BufferLength** es cero, no se hace referencia a `Buffer` ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica `CompCode`.

Si `CompCode` es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si `CompCode` es MQCC_WARNING:

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838 ') Opciones de informe en descriptor de mensaje no reconocidas.

Si `CompCode` es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') La cola base de alias no es un tipo válido.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') recurso de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') La comprobación de autorización de la estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_ERROR
(2349, X'92D') La estructura del recurso de acoplamiento no es válida.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR
(2416, X' 970 ') La estructura de parámetro de grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR
(2235, X'8BB') La estructura de cabecera PCF no es válida.

MQRC_CFIF_ERROR
(2414, X'96E') La estructura del parámetro de filtro de enteros PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR
(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR
(2237, X'8BD') Estructura de parámetro de entero PCF o estructura de parámetro de entero PCIF*64 no válida.

MQRC_CFSF_ERROR
(2415, X'96F') La estructura del parámetro de filtro de serie PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR
(2238, X'8BE') La estructura de parámetro de lista de series PCF no es válida.

MQRC_CFST_ERROR
(2239, X'8BF') La estructura del parámetro de serie PCF no es válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

ERROR DE MQRC_CONTENT_
2554 (X'09FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje se puede entregar a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La cola de transmisión predeterminada no es local.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

MQRC_DH_ERROR
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

MQRC_DLH_ERROR
(2141, X'85D') La estructura de cabecera de letra muerda no es válida.

MQRC_EPH_ERROR
(2420, X' 974 ') La estructura PCF incorporada no es válida.

MQRC_EXPIRY_ERROR
(2013, X'7DD') Tiempo de caducidad no válido.

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Código de comentarios no válido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HEADER_ERROR
(2142, X'85E') IBM MQ estructura de cabecera no válida.

MQRC_IIH_ERROR
(2148, X'864 ') IMS estructura de cabecera de información no válida.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Falta la cola de respuestas.

MQRC_MISSING_WIH
(2332, X'91C') Los datos de mensaje no empiezan por MQWIH.

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Los distintivos de mensaje no son válidos.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE
(2042, X'7FA') El objeto ya está abierto con opciones en conflicto.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X' 938 ') Nivel de objeto no compatible.

MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nombre de objeto no válido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Objeto no exclusivo.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nombre de gestor de colas de objeto no válido.

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Los registros de objeto no son válidos.

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR
(2044, X'7FC') La estructura de descriptor de objeto no es válida.

ERROR DE MQRC_OFFSET_
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR
(2149, X'865 ') Estructuras PCF no válidas.

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

MQRC_PRIORITY_ERROR
(2050, X'802 ') La prioridad del mensaje no es válida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_Q_TYPE_ERROR
(2057, X'809 ') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') El nombre de cola remota no es válido.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_RFH_ERROR
(2334, X'91E') La estructura MQRFH o MQRFH2 no es válida.

MQRC_RMH_ERROR
(2220, X'8AC') La estructura de cabecera de mensaje de referencia no es válida.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si se envía la publicación al suscriptor.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

ERROR MQRC_TM
(2265, X'8D9') La estructura de mensajes del desencadenante no es válida.

ERROR MQRC_TMC
(2191, X'88F') La estructura del mensaje desencadenante de caracteres no es válida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Cola base alias desconocida.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nombre de objeto desconocido.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestor de colas de objetos desconocido.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestor de colas remoto desconocido.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Cola de transmisión desconocida.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR
(2333, X'91D') La estructura MQWIH no es válida.

MQRC_ERR_CF_LEVEL
(2366, X'93E') La estructura del recurso de acoplamiento tiene un nivel incorrecto.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Cola de transmisión con un uso incorrecto.

MQRC_XQH_ERROR
(2260, X'8D4') La estructura de cabecera de cola de transmisión no es válida.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias:
   - Utilice la llamada MQPUT para colocar varios mensajes en la *misma* cola.

     En primer lugar, se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .
   - Utilice la llamada MQPUT1 para colocar sólo *un* mensaje en una cola.

     Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.
2. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Sin embargo, en la mayoría de los entornos, la llamada MQPUT1 no satisface estas condiciones, por lo que no conserva el orden de los mensajes. En su lugar, se debe utilizar la llamada MQPUT en estos entornos. Consulte Notas de uso de MQPUT para obtener detalles.
3. La llamada MQPUT1 se puede utilizar para transferir mensajes a listas de distribución. Para obtener información general sobre esto, consulte las notas de uso para las llamadas MQOPEN y MQPUT.

   Las listas de distribución están soportadas en los entornos siguientes:
   - AIX
   - IBM i
   - Linux
   - Solaris
   - Windows

   y para clientes de IBM MQ conectados a estos sistemas.

Las diferencias siguientes se aplican cuando se utiliza la llamada MQPUT1 :

a. Si la aplicación proporciona registros de respuesta MQRR, se deben proporcionar utilizando la estructura MQOD; no se pueden proporcionar utilizando la estructura MQPMO.

b. MQPUT1 nunca devuelve el código de razón MQRC_OPEN_FAILED en los registros de respuesta; si una cola no se puede abrir, el registro de respuesta para dicha cola contiene el código de razón resultante de la operación de apertura.

   Si una operación abierta para una cola se ejecuta correctamente con un código de terminación de MQCC_WARNING, el código de terminación y el código de razón en el registro de respuesta para dicha cola se sustituyen por los códigos de terminación y razón resultantes de la operación de transferencia.

   Al igual que con las llamadas MQOPEN y MQPUT, el gestor de colas establece los registros de respuesta (si se proporcionan) sólo cuando el resultado de la llamada no es el mismo para todas las colas de la lista de distribución; esto se indica mediante la finalización de la llamada con el código de razón MQRC_MULTIPLE_REASON.

4. Si se utiliza la llamada MQPUT1 para colocar un mensaje en una cola de clúster, la llamada se comporta como si se hubiera especificado MQOO_BIND_NOT_FIXED en la llamada MQOPEN.
5. Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Para obtener más información sobre esto, consulte las notas de uso para la llamada MQPUT.
6. Si se produce más de una de las situaciones de aviso (consulte el parámetro **CompCode**), el código de razón devuelto es el primero de la lista siguiente que se aplica:

   a. MQRC_MULTIPLE_RAZONES

   b. MQRC_INCOMPLETE_MSG

   c. MQRC_INCOMPLETE_GROUP

   d. MQRC_PRIORITY_EXCEEDS_MAXIMUM o MQRC_UNKNOWN_REPORT_OPTION
7. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

   - Si el tamaño del parámetro **Buffer** es menor que la longitud especificada por el parámetro **BufferLength** , la llamada falla con el código de razón MQRC_BUFFER_LENGTH_ERROR.
   - El parámetro **Buffer** se declara como de tipo `String`. Si los datos que se van a colocar en la cola no son del tipo `String`, utilice elMQPUT1Any en lugar de MQPUT1.

     La llamada MQPUT1Any tiene los mismos parámetros que la llamada MQPUT1 , excepto que el parámetro **Buffer** se declara como de tipo Any, lo que permite colocar cualquier tipo de datos en la cola. Sin embargo, esto significa que `Buffer` no se puede comprobar para asegurarse de que tiene como mínimo `BufferLength` bytes de tamaño.
8. Cuando se emite una llamada MQPUT1 con MQPMO_SYNCPOINT, el comportamiento predeterminado cambia, de modo que la operación put se completa de forma asíncrona. Esto puede crear cambio de comportamiento en algunas aplicaciones que se basan en la devolución de determinados campos de las estructuras MQOD y MQMD y que ahora contienen valores no definidos. Una aplicación puede especificar MQPMO_SYNC_RESPONSE para asegurarse de que la operación de colocación se realiza de forma síncrona y de que se han completado todos los valores de campo adecuados.

Invocación en C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
       BufferLength, Buffer, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQOD     ObjDesc;       /* Object descriptor */
```

```
MQMD     MsgDesc;       /* Message descriptor */
MQPMO    PutMsgOpts;    /* Options that control the action of MQPUT1 */
MQLONG   BufferLength;  /* Length of the message in Buffer */
MQBYTE   Buffer[n];     /* Message data */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                         BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object descriptor
 01  OBJDESC.
     COPY CMQODV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQPUT1
 01  PUTMSGOPTS.
     COPY CMQPMOV.
**   Length of the message in BUFFER
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Message data
 01  BUFFER        PIC X(n).
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl ObjDesc       like MQOD;      /* Object descriptor */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl PutMsgOpts    like MQPMO;     /* Options that control the action of
                                     MQPUT1 */
dcl BufferLength  fixed bin(31);  /* Length of the message in Buffer */
dcl Buffer        char(n);        /* Message data */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
         CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,   X
               BUFFER,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN         DS       F      Connection handle
OBJDESC       CMQODA   ,      Object descriptor
MSGDESC       CMQMDA   ,      Message descriptor
PUTMSGOPTS    CMQPMOA  ,      Options that control the action of MQPUT1
BUFFERLENGTH  DS       F      Length of the message in BUFFER
BUFFER        DS       CL(n)  Message data
```

```
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Invocación en Visual Basic

Windows

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn        As Long   'Connection handle'
Dim ObjDesc      As MQOD   'Object descriptor'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim PutMsgOpts   As MQPMO  'Options that control the action of MQPUT1'
Dim BufferLength As Long   'Length of the message in Buffer'
Dim Buffer       As String 'Message data'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQSET - Establecer atributos de objeto

Utilice la llamada MQSET para cambiar los atributos de un objeto representado por un descriptor de contexto. El objeto debe ser una cola.

Sintaxis

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectores*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de Hconn ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

z/OS En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este descriptor de contexto representa el objeto de cola con atributos que se van a establecer. El descriptor de contexto ha sido devuelto por una llamada MQOPEN anterior que especificaba la opción MQOO_SET.

SelectorCount

Tipo: MQLONG - entrada

Este es el recuento de selectores que se proporcionan en la matriz `Selectors` . Es el número de atributos que se van a establecer. Cero es un valor válido. El número máximo permitido es 256.

Selectores

Tipo: MQLONGxSelectorRecuento-entrada

Esta es una matriz de selectores de atributos **`SelectorCount`** ; cada selector identifica un atributo (entero o carácter) con un valor que se va a establecer.

Cada selector debe ser válido para el tipo de cola que representa `Hobj` . Sólo se permiten determinados valores MQIA_ * y MQCA_ *; tal como se indica más adelante.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectores MQIA_ *) deben especificarse en `IntAttrs` en el mismo orden en el que aparecen estos selectores en `Selectors`. Los valores de atributo que corresponden a selectores de atributo de carácter (selectores MQCA_ *) deben especificarse en `CharAttrs` en el mismo orden en el que se producen dichos selectores. Los selectores MQIA_ * se pueden intercalar con los selectores MQCA_ *; sólo es importante el orden relativo dentro de cada tipo.

Puede especificar el mismo selector más de una vez; si lo hace, el último valor especificado para un selector determinado es el que entra en vigor.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores MQIA_ * residen dentro del rango de MQIA_FIRST a MQIA_LAST, y los selectores MQCA_ * dentro del rango de MQCA_FIRST a MQCA_LAST.

   Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USED definen el valor más alto que acepta el gestor de colas.
2. Si todos los selectores MQIA_ * aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices `Selectors` y `IntAttrs` .
3. Si el parámetro **`SelectorCount`** es cero, no se hace referencia a `Selectors` ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

Los atributos que se pueden establecer se listan en la tabla siguiente. No se pueden establecer otros atributos utilizando esta llamada. Para los selectores de atributos MQCA_ *, la constante que define la longitud en bytes de la serie necesaria en `CharAttrs` se proporciona entre paréntesis.

Tabla 554. Selectores de atributos MQSET para colas

Selector	Descripción	Nota
MQCA_TRIGGER_DATA	Datos de desencadenante (MQ_TRIGGER_DATA_LENGTH).	
MQIA_DIST_LISTS	Soporte de lista de distribución.	1
INHIBIDORES DE MQIA_GET	Si se permiten operaciones get.	
INHIBIDORES DE MQIA_PUT	Si se permiten las operaciones de colocación.	
MQIA_TRIGGER_CONTROL	Control de desencadenante.	
MQIA_PROFUNDIDAD	Profundidad del desencadenante.	
MQIA_TRIGGER_MSG_PRIORITY	Umbral de prioridad del mensaje para activaciones.	
MQIA_TIPO_TRIGGER_TYPE	Tipo de desencadenante.	

Nota:

1. Soportado sólo en las plataformas siguientes:
   - AIX AIX
   - IBM i IBM i
   - Linux Linux
   - Solaris Solaris

- Windows Windows

  y para IBM MQ MQI clients conectados a estos sistemas.

IntAttrCount
Tipo: MQLONG - entrada

Es el número de elementos de la matriz `IntAttrs` y debe ser como mínimo el número de selectores MQIA_ * en el parámetro **`Selectors`** . Cero es un valor válido si no hay ninguno.

IntAttrs
Tipo: MQLONGxIntAttrCount -entrada

Esta es una matriz de valores de atributo de entero de `IntAttrCount` . Estos valores de atributo deben estar en el mismo orden que los selectores MQIA_ * de la matriz `Selectors` .

Si el parámetro **`IntAttrCount`** o **`SelectorCount`** es cero, no se hace referencia a `IntAttrs` ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

CharAttrLongitud
Tipo: MQLONG - entrada

Esta es la longitud en bytes del parámetro **`CharAttrs`** y debe ser como mínimo la suma de las longitudes de los atributos de caracteres especificados en la matriz `Selectors` . Cero es un valor válido si no hay selectores MQCA_ * en `Selectors`.

CharAttrs
Tipo: MQCHAR x CharAttrLongitud-entrada

Es el almacenamiento intermedio que contiene los valores de atributo de carácter, concatenados entre sí. La longitud del búfer se proporciona en el parámetro **`CharAttrLength`**.

Los atributos de caracteres deben especificarse en el mismo orden que los selectores MQCA_ * en la matriz `Selectors` . La longitud de cada atributo de carácter es fija (consulte Selectores). Si el valor que se va a establecer para un atributo contiene menos caracteres no en blanco que la longitud definida del atributo, rellene el valor de `CharAttrs` a la derecha con espacios en blanco para que el valor del atributo coincida con la longitud definida del atributo.

Si el parámetro **`CharAttrLength`** o **`SelectorCount`** es cero, no se hace referencia a `CharAttrs` ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica `CompCode`.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si `CompCode` es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Longitud de atributos de caracteres no válida.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Serie de atributos de carácter no válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INHIBIDOR_VALOR_ERROR
(2020, X'7E4') El valor del atributo de cola con inhibición-get o con inhibición-put no es válido.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Recuento de atributos enteros no válidos.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Matriz de atributos enteros no válida.

MQRC_NOT_OPEN_FOR_SET
(2040, X'7F8') Cola no abierta para el conjunto.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811 ') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR
(2067, X'813 ') El selector de atributos no es válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812 ') Recuento de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_TRIGGER_CONTROL_ERROR
(2075, X'81B') El valor del atributo trigger-control no es válido.

MQRC_TRIGGER_DEPTH_ERROR
(2076, X'81C') El valor del atributo de profundidad de desencadenante no es válido.

MQRC_TRIGGER_MSG_PRIORITY_ERR
(2077, X'81D') El valor del atributo trigger-message-priority no es válido.

MQRC_TRIGGER_TYPE_ERROR
(2078, X'81E') El valor del atributo de tipo desencadenante no es válido.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Utilizando esta llamada, la aplicación puede especificar una matriz de atributos enteros, o una colección de series de atributos de caracteres, o ambos. Si no se producen errores, todos los atributos especificados se establecen simultáneamente. Si se produce un error (por ejemplo, si un selector no es válido o se intenta establecer un atributo en un valor que no es válido), la llamada falla y no se establece ningún atributo.
2. Los valores de los atributos se pueden determinar utilizando la llamada MQINQ; consulte “MQINQ-Consultar atributos de objeto” en la página 715 para obtener detalles.

   Nota: No todos los atributos con valores que se pueden consultar utilizando la llamada MQINQ pueden cambiar sus valores utilizando la llamada MQSET. Por ejemplo, no se pueden establecer atributos de objeto de proceso o de gestor de colas con esta llamada.
3. Los cambios de atributo se conservan entre los reinicios del gestor de colas (aparte de las alteraciones en las colas dinámicas temporales, que no sobreviven a los reinicios del gestor de colas).
4. No puede cambiar los atributos de una cola modelo utilizando la llamada MQSET. Sin embargo, si abre una cola modelo utilizando la llamada MQOPEN con la opción MQOO_SET, puede utilizar la llamada MQSET para establecer los atributos de la cola local dinámica creada por la llamada MQOPEN.
5. Si el objeto que se está definiendo es una cola de clúster, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria.

Para obtener más información sobre los atributos de objeto, consulte:

- “Atributos para colas” en la página 848
- “Atributos de las listas de nombres” en la página 883
- “Atributos de las definiciones de proceso” en la página 885
- “Atributos para el gestor de colas” en la página 810

Invocación en C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
MQLONG   Selectors[n];    /* Array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
MQLONG   IntAttrs[n];     /* Array of integer attributes */
MQLONG   CharAttrLength;  /* Length of character attributes buffer */
MQCHAR   CharAttrs[n];    /* Character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                   INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                   CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN            PIC S9(9) BINARY.
**   Object handle
 01  HOBJ             PIC S9(9) BINARY.
**   Count of selectors
 01  SELECTORCOUNT    PIC S9(9) BINARY.
**   Array of attribute selectors
 01  SELECTORS-TABLE.
  02  SELECTORS        PIC S9(9) BINARY OCCURS n TIMES.
**   Count of integer attributes
 01  INTATTRCOUNT     PIC S9(9) BINARY.
**   Array of integer attributes
 01  INTATTRS-TABLE.
  02  INTATTRS         PIC S9(9) BINARY OCCURS n TIMES.
**   Length of character attributes buffer
 01  CHARATTRLENGTH   PIC S9(9) BINARY.
**   Character attributes
 01  CHARATTRS        PIC X(n).
**   Completion code
 01  COMPCODE         PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON           PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn           fixed bin(31);  /* Connection handle */
dcl Hobj            fixed bin(31);  /* Object handle */
dcl SelectorCount   fixed bin(31);  /* Count of selectors */
dcl Selectors(n)    fixed bin(31);  /* Array of attribute selectors */
dcl IntAttrCount    fixed bin(31);  /* Count of integer attributes */
dcl IntAttrs(n)     fixed bin(31);  /* Array of integer attributes */
dcl CharAttrLength  fixed bin(31);  /* Length of character attributes
                                       buffer */
dcl CharAttrs       char(n);        /* Character attributes */
dcl CompCode        fixed bin(31);  /* Completion code */
dcl Reason          fixed bin(31);  /* Reason code qualifying
                                       CompCode */
```

Invocación en ensamblador de alto nivel

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

Declare los parámetros como se indica a continuación:

```
HCONN           DS  F      Connection handle
HOBJ            DS  F      Object handle
SELECTORCOUNT   DS  F      Count of selectors
SELECTORS       DS  (n)F   Array of attribute selectors
INTATTRCOUNT    DS  F      Count of integer attributes
INTATTRS        DS  (n)F   Array of integer attributes
CHARATTRLENGTH  DS  F      Length of character attributes buffer
CHARATTRS       DS  CL(n)  Character attributes
COMPCODE        DS  F      Completion code
REASON          DS  F      Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn          As Long   'Connection handle'
Dim Hobj           As Long   'Object handle'
Dim SelectorCount  As Long   'Count of selectors'
Dim Selectors      As Long   'Array of attribute selectors'
Dim IntAttrCount   As Long   'Count of integer attributes'
Dim IntAttrs       As Long   'Array of integer attributes'
Dim CharAttrLength As Long   'Length of character attributes buffer'
Dim CharAttrs      As String 'Character attributes'
Dim CompCode       As Long   'Completion code'
Dim Reason         As Long   'Reason code qualifying CompCode'
```

MQSETMP-Establecer propiedad de mensaje

Utilice la llamada MQSETMP para establecer o modificar una propiedad de un descriptor de mensaje.

Sintaxis

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** . Si el descriptor de contexto de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra estableciendo una propiedad del descriptor de contexto de mensaje; de lo contrario, la llamada falla con el código de razón MQRC_CONNECTION_BROKEN.

Hmsg
Tipo: MQHMSG-entrada

Este es el manejador de mensajes que se va a modificar. El valor ha sido devuelto por una llamada MQCRTMH anterior.

SetPropOpts
Tipo: MQSMPO-entrada

Controlar cómo se establecen las propiedades de mensaje.

Esta estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de mensaje. La estructura es un parámetro de entrada en la llamada MQSETMP. Consulte MQSMPO para obtener más información.

Nombre
Tipo: MQCHARV-entrada

Es el nombre de la propiedad que se va a establecer.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PropDesc
Tipo: MQPD-entrada/salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo:

- qué sucede si la propiedad no está soportada
- a qué contexto de mensaje pertenece la propiedad
- en qué mensajes se copia la propiedad a medida que fluye

Consulte MQPD para obtener más información sobre esta estructura.

Tipo
Tipo: MQLONG - entrada

El tipo de datos de la propiedad que se está definiendo. Puede ser uno de los siguientes:

MQTYPE_BOOLEAN
Un valor booleano. *ValueLength* debe ser 4.

MQTYPE_BYTE_STRING
Una serie de bytes. *ValueLength* debe ser cero o mayor.

MQTYPE_INT8
Un entero con signo de 8 bits. *ValueLength* debe ser 1.

MQTYPE_INT16
Un entero con signo de 16 bits. *ValueLength* debe ser 2.

MQTYPE_INT32
Un entero con signo de 32 dígitos. *ValueLength* debe ser 4.

MQTYPE_INT64
Un entero con signo de 64 bits. *ValueLength* debe ser 8.

MQTYPE_FLOAT32
Un número de coma flotante de 32 bits. *ValueLength* debe ser 4.

Nota: este tipo no está soportado con aplicaciones que utilizan IBM COBOL for z/OS.

MQTYPE_FLOAT64
Un número de coma flotante de 64 bits. *ValueLength* debe ser 8.

Nota: este tipo no está soportado con aplicaciones que utilizan IBM COBOL for z/OS.

MQTYPE_STRING
Una serie de caracteres. *ValueLength* debe ser cero o mayor, o el valor especial MQVL_NULL_TERMINATED.

MQTYPE_NULL
La propiedad existe pero tiene un valor nulo. *ValueLength* debe ser cero.

ValueLength
Tipo: MQLONG - entrada

Longitud en bytes del valor de propiedad en el parámetro *Valor* . Cero sólo es válido para valores nulos o para series o series de bytes. Cero indica que la propiedad existe pero que el valor no contiene caracteres ni bytes.

El valor debe ser mayor o igual que cero o el siguiente valor especial si el parámetro *Tipo* tiene establecido MQTYPE_STRING:

MQVL_NULL_TERMINATED
El valor está delimitado por el primer nulo encontrado en la serie. El valor nulo no se incluye como parte de la serie. Este valor no es válido si no se ha establecido también MQTYPE_STRING.

Nota: El carácter nulo utilizado para terminar una serie si se establece MQVL_NULL_TERMINATED es un valor nulo del juego de caracteres del valor.

Valor
Tipo: MQBYTExValueLongitud-entrada

El valor de la propiedad que se va a establecer. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del valor.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si *ValueLength* es cero, no se hace referencia a *Valor* . En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de valor no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PD_ERROR
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

ERROR_TIPO_PROPIEDAD_MQRC
(2473, X'09A9') Tipo de datos de propiedad no válido.

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

MQRC_SMPO_ERROR
(2463, X'099F') Establecer estructura de opciones de propiedad de mensaje no válida.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Invocación en C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;       /* Connection handle */
MQHMSG   Hmsg;        /* Message handle */
MQSMPO   SetPropOpts; /* Options that control the action of MQSETMP */
MQCHARV  Name;        /* Property name */
MQPD     PropDesc;    /* Property descriptor */
MQLONG   Type;        /* Property data type */
MQLONG   ValueLength; /* Length of property value in Value */
MQBYTE   Value[n];    /* Property value */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDESC, TYPE,
                          VALUELENGTH, VALUE, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Message handle
 01  HMSG    PIC S9(18) BINARY.
**   Options that control the action of MQSETMP
 01  SETMSGOPTS.
     COPY CMQSMPOV.
**   Property name
 01  NAME
    COPY CMQCHRVV.
**   Property descriptor
 01  PROPDESC.
     COPY CMQPDV.
**   Property data type
 01  TYPE          PIC S9(9) BINARY.
**   Length of property value in VALUE
 01  VALUELENGTH   PIC S9(9) BINARY.
**   Property value
 01  VALUE         PIC X(n).
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
              Value, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl SetPropOpts like MQSMPO;    /* Options that control the action of MQSETMP */
dcl Name        like MQCHARV;   /* Property name */
dcl PropDesc    like MQPD;      /* Property descriptor */
dcl Type        fixed bin(31);  /* Property data type */
dcl ValueLength fixed bin(31);  /* Length of property value in Value */
dcl Value       char(n);        /* Property value */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
         CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
               VALUE,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
SETMSGOPTS   CMQSMPOA   ,     Options that control the action of MQSETMP
NAME         CMQCHRVA   ,     Property name
PROPDESC     CMQPDA     ,     Property descriptor
TYPE         DS         F     Property data type
VALUELENGTH  DS         F     Length of property value in VALUE
VALUE        DS         CL(n) Property value
```

```
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQSTAT-Recuperar información de estado

Utilice la llamada MQSTAT para recuperar información de estado. El tipo de información de estado devuelta viene determinado por el valor de tipo especificado en la llamada.

Sintaxis

MQSTAT (*Hconn*, *Tipo*, *Stat*, *Compcode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Tipo

Tipo: MQLONG - entrada

Tipo de información de estado que se está solicitando. Los > valores válidos son:

MQSTAT_TYPE_ASYNC_ERROR
Devuelve información sobre operaciones de transferencia asíncronas anteriores.

MQSTAT_TYPE_RECONNECTION

Devuelve información sobre la reconexión. Si la conexión se está reconectando o no se ha podido volver a conectar, la información describe la anomalía que ha hecho que la conexión empezara a reconectarse.

Este valor sólo es válido para conexiones de cliente. Para otros tipos de conexión, la llamada falla con el código de razón **MQRC_ENVIRONMENT_ERROR**

MQSTAT_TYPE_RECONNECTION_ERROR

Devuelve información sobre una anomalía anterior relacionada con la reconexión. Si la conexión no se ha podido reconectar, la información describe la anomalía que ha hecho que la reconexión fallara.

Este valor sólo es válido para conexiones de cliente. Para otros tipos de conexión, la llamada falla con el código de razón **MQRC_ENVIRONMENT_ERROR**.

Estado

Tipo: MQSTS-entrada/salida

Estructura de información de estado. Consulte “MQSTS-Estructura de informes de estado” en la página 597 para obtener los detalles.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR
(2374, X' 946 ') La salida de API ha fallado

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_Q_MGR_STOPPING
(2162,X'872'-Se está deteniendo el gestor de colas

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STAT_TYPE_ERROR
(2430, X'97E' Error con tipo MQSTAT

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_STS_ERROR
(2426, X'97A') Error con estructura MQSTS

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

1. Una llamada a `MQSTAT` especificando un tipo de `MQSTAT_TYPE_ASYNC_ERROR` devuelve información sobre operaciones `MQPUT` y `MQPUT1` asíncronas anteriores. La estructura `MQSTS` que se devuelve de la llamada `MQSTAT` contiene la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de `MQCC_WARNING`, en su lugar se devuelve una anomalía posterior con un código de terminación de `MQCC_FAILED` .
2. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a `MQSTAT` , se devuelve un `CompCode` de `MQCC_OK` y una razón de `MQRC_NONE` en la estructura `MQSTS` .
3. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven mediante tres campos de contador: `PutSuccessCount`, `PutWarningCount` y `PutFailureCount`. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en

lugar de una vez por lista de distribución). Un contador no se incrementa más allá del valor positivo máximo AMQ_LONG_MAX.

4. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.
5. El comportamiento de MQSTAT depende del valor del parámetro **MQSTAT Type** que proporcione.
6. **MQSTAT_TYPE_ASYNC_ERROR**
   a. Una llamada a MQSTAT especificando un tipo de MQSTAT_TYPE_ASYNC_ERROR devuelve información sobre operaciones MQPUT y MQPUT1 asíncronas anteriores. La estructura MQSTS que se devuelve de la llamada MQSTAT contiene la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de MQCC_WARNING, en su lugar se devuelve una anomalía posterior con un código de terminación de MQCC_FAILED .
   b. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT , se devuelve un CompCode de MQCC_OK y una razón de MQRC_NONE en la estructura MQSTS .
   c. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven mediante tres campos de contador: PutSuccessCount, PutWarningCount y PutFailureCount. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en lugar de una vez por lista de distribución). Un contador no se incrementa más allá del valor positivo máximo AMQ_LONG_MAX.
   d. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.

MQSTAT_TYPE_RECONNECTION

Supongamos que llama a MQSTAT con Type establecido en MQSTAT_TYPE_RECONNECTION dentro de un manejador de sucesos durante la reconexión. Considere estos ejemplos.

El cliente está intentando la reconexión o no se ha podido volver a conectar.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason puede ser MQRC_CONNECTION_BROKEN o MQRC_Q_MGR_QUIESCING. ObjectType es MQOT_Q_MGR, ObjectName es el nombre del gestor de colas y ObjectQMgrName está en blanco.

El cliente ha completado la reconexión correctamente o nunca se ha desconectado.
CompCode en la estructura MQSTS es MQCC_OK y Reason es MQRC_NONE

Las llamadas posteriores a MQSTAT devuelven los mismos resultados.

MQSTAT_TYPE_RECONNECTION_ERROR

Supongamos que llama a MQSTAT con Type establecido en MQSTAT_TYPE_RECONNECTION_ERROR en respuesta a la recepción de MQRC_RECONNECT_FAILED a una llamada MQI. Considere estos ejemplos.

Se ha producido un error de autorización cuando se estaba reabriendo una cola durante la reconexión con un gestor de colas diferente.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason es la razón por la que la reconexión ha fallado, como por ejemplo MQRC_NOT_AUTHORIZED. ObjectType es el tipo de objeto que ha causado el problema, como por ejemplo MQOT_QUEUE, ObjectName es el nombre de la cola y ObjectQMgrName el nombre del gestor de colas propietario de la cola.

Se ha producido un error de conexión de socket durante la reconexión.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason es la razón por la que la reconexión ha fallado, como por ejemplo MQRC_HOST_NOT_AVAILABLE. ObjectType es MQOT_Q_MGR, ObjectName es el nombre del gestor de colas y ObjectQMgrName está en blanco.

Las llamadas posteriores a MQSTAT devuelven los mismos resultados.

Invocación en C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;        /* Connection Handle */
MQLONG StatType;    /* Status type */
MQSTS Stat;           /* Status information structure */
MQLONG CompCode;    /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**    Connection handle
 01    HCONN      PIC S9(9)      BINARY.
**    Status type
 01    STATTYPE   PIC S9(9)      BINARY.
**    Status information
 01    STAT.
    COPY CMQSTSV.
**    Completion code
 01    COMPCODE   PIC S9(9)      BINARY.
**    Reason code qualifying COMPCODE
 01    REASON     PIC S9(9)      BINARY.
```

Invocación en PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31); /* Connection handle */
dcl StatType    fixed bin(31); /* Status type */
dcl Stat        like MQSTS;    /* Status information structure */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason      fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación de System/390 Assembler

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS     F    Connection handle
STATTYPE  DS     F    Status type
STAT      CMQSTSA,    Status information structure
COMPCODE  DS     F    Completion code
REASON    DS     F    Reason code qualifying COMPCODE
```

MQSUB - Registrar suscripción

Utilice la llamada MQSUB para registrar la suscripción de aplicaciones a un tema determinado.

Sintaxis

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

SubDesc

Tipo: MQSD - entrada/salida

Esta es una estructura que identifica el objeto en uso que está registrando la aplicación. Consulte "MQSD - Descriptor de suscripción" en la página 572 para obtener más información.

Hobj

Tipo: MQHOBJ - entrada/salida

Este descriptor de contexto representa el acceso que se ha establecido para obtener los mensajes enviados a esta suscripción. Estos mensajes se pueden almacenar en una cola específica o el gestor de colas puede gestionar su almacenamiento sin utilizar ninguna cola específica.

Para utilizar una cola específica, debe asociarla con la suscripción al crear la suscripción. Puede hacerlo de dos maneras:

- Utilizando el mandato DEFINE SUB MQSC y proporcionando a ese mandato el nombre de un objeto de cola.
- Proporcionando este descriptor de contexto al llamar a MQSUB con MQSO_CREATE

  Si se proporciona este descriptor de contexto como un parámetro de entrada en la llamada, debe ser un descriptor de contexto de objeto válido desvuelto de una llamada MQOPEN anterior de una cola utilizando al menos una de las opciones siguientes:

  – MQOO_INPUT_*
  – MQOO_BROWSE
  – MQOO_OUTPUT (si la cola es una cola remota)

  Si este no es el caso, la llamada falla con MQRC_HOBJ_ERROR. No puede ser un descriptor de contexto de objeto a una cola de alias que se resuelva en un objeto de tema. Si es así, la llamada falla con MQRC_HOBJ_ERROR.

Si el gestor de colas va a gestionar el almacenamiento de los mensajes enviados a esta suscripción, se debe establecer al crear la suscripción, utilizando la opción MQSO_MANAGED. A continuación, el gestor de colas devuelve este descriptor de contexto como un parámetro de salida en la llamada. El descriptor de contexto devuelto se conoce como descriptor de contexto gestionado. Si se especifica MQHO_NONE pero no se especifica MQSO_MANAGED, la llamada falla con MQRC_HOBJ_ERROR.

Cuando el gestor de colas le devuelve un descriptor de contexto gestionado, puede utilizarlo en una llamada MQGET o MQCB con o sin opciones browse, en una llamada MQINQ, o en MQCLOSE. No puede utilizarlo en MQPUT, MQSUB, MQSET; si se intenta, fallará con MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR o MQRC_NOT_OPEN_FOR_SET.

Si esta suscripción se reanuda utilizando la opción MQSO_RESUME en la estructura MQSD, el descriptor de contexto se puede devolver a la aplicación en este parámetro estableciendo MQSO_MANAGED en MQHO_NONE. Puede hacer esto independientemente de si la suscripción utiliza o no un descriptor de contexto gestionado, y puede ser útil para proporcionar suscripciones creadas

utilizando DEFINE SUB con el descriptor de contexto a la cola de suscripciones definida en ese mandato. En el caso donde se está reanudando una suscripción creada administrativamente, la cola se abre con MQOO_INPUT_AS_Q_DEF y MQOO_BROWSE. Si necesita especificar otras opciones, la aplicación debe abrir la cola de suscripciones explícitamente y proporcionar el descriptor de contexto de objeto en la llamada. Si hay un problema al abrir la cola, la llamada falla con MQRC_INVALID_DESTINATION. Si se proporciona *Hobj*, debe ser equivalente a *Hobj* en la llamada MQSUB original. Esto significa que si se proporciona un descriptor de contexto de objeto devuelto de una llamada MQOPEN, el descriptor de contexto debe ser a la misma cola que se ha utilizado anteriormente. Si no es la misma cola, la llamada falla con MQRC_HOBJ_ERROR.

Si esta suscripción se está modificando utilizando la opción MQSO_ALTER en la estructura MQSD, se puede proporcionar un *Hobj* distinto. Las publicaciones entregadas a la cola e identificadas anteriormente mediante este parámetro permanecen en esa cola y es responsabilidad de la aplicación recuperar estos mensajes si ahora el parámetro **Hobj** representa una cola distinta.

Tabla 555. Utilización de hobj con varias opciones de suscripción

Opciones	***Hobj***	**Descripción**
MQSO_CREATE + MQSO_MANAGED	Ignorado en la salida	Crea una suscripción con almacenamiento de mensajes gestionado por el gestor de colas
MQSO_CREATE	Un descriptor de contexto de objeto válido	Crea una suscripción que facilita una cola específica como destino de los mensajes.
MQSO_RESUME	MQHO_NONE	Reanuda una suscripción creada anteriormente independientemente de si está gestionada o no, y el gestor de colas devuelve el descriptor de contexto de objeto para que lo utilice la aplicación.
MQSO_RESUME	Un descriptor de contexto de objeto coincidente válido	Reanuda una suscripción creada anteriormente que utiliza una cola específica como destino de los mensajes y utiliza un descriptor de contexto de objeto con opciones de apertura específicas.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Modifica una suscripción existente que utilizaba anteriormente una cola específica, por lo que ahora es una suscripción gestionada. La clase de destino (gestionado o no) no se puede cambiar.
MQSO_ALTER	Un descriptor de contexto de objeto válido	Modifica una suscripción existente, gestionada o no, para que ahora utilice una cola específica. Cuando no se utiliza la opción MQSO_MANAGED, la cola proporcionada se puede cambiar, pero no se puede cambiar la clase de destino (gestionada o no).

Independientemente de si se ha proporcionado o devuelto, se debe especificar *Hobj* en las llamadas MQGET o MQCB subsiguientes que deseen recibir los mensajes de publicación enviados a esta suscripción.

El descriptor de contexto *Hobj* ya no es válido cuando se emite en él la llamada MQCLOSE, o cuando la unidad de proceso que define el ámbito del descriptor de contexto termina (hasta que la aplicación se desconecta). El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Consulte Hconn (MQHCONN) - salida para obtener información sobre el ámbito del descriptor de contexto. Un MQCLOSE del descriptor de contexto *Hobj* no afecta al descriptor de contexto *Hsub*.

Hsub
Tipo: MQHOBJ - salida

Este descriptor de contexto representa la suscripción que se ha realizado. Se puede utilizar para otras dos operaciones:

- Se puede utilizar en una llamada MQSUBRQ subsiguiente para solicitar que las publicaciones se envíen cuando se haya utilizado la opción MQSO_PUBLICATIONS_ON_REQUEST al realizar la suscripción.
- Se puede utilizar en una llamada MQCLOSE subsiguiente para eliminar la suscripción que se ha realizado. El descriptor de contexto *Hsub* deja de ser válido cuando se emite la llamada MQCLOSE, o cuando la unidad de proceso que define el ámbito del descriptor de contexto termina. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Un MQCLOSE del descriptor de contexto *Hsub* no afecta al descriptor de contexto *Hobj*.

Este descriptor de contexto no se puede pasar a una llamada MQGET o MQCB. Debe utilizar el parámetro **Hobj**. No puede utilizar este descriptor de contexto en ninguna llamada IBM MQ que no sea MQCLOSE o MQSUBRQ. Si se pasa este descriptor de contexto a cualquier otra llamada IBM MQ, se genera MQRC_HOBJ_ERROR.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón
Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK, el código de razón es el siguiente:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED, el código de razón es uno de los siguientes:

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_DURABILITY_NOT_ALLOWED
2436 (X'0984') Ha fallado una llamada MQSUB utilizando la opción MQSO_DURABLE.

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_HOBJ_ERROR
2019 (X'07E3') Descriptor de contexto de objeto Hobj no válido.

MQRC_IDENTITY_MISMATCH
2434 (X'0982') El nombre de suscripción coincide con una suscripción existente.

MQRC_NOT_AUTHORIZED
2035 (X'07F3') El usuario no está autorizado para realizar la operación.

MQRC_NO_SUBSCRIPTION
2428 (X'097C') El nombre de suscripción identificado no existe.

MQRC_OBJECT_STRING_ERROR
2441 (X'0989') El campo Objectstring no es válido.

MQRC_OPTIONS_ERROR
2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR_QUIESCING
2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_RECONNECT_Q_MGR_REQD
2555 (X'09FB'X) Se requiere la opción MQCNO_RECONNECT_Q_MGR.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

MQRC_SD_ERROR
2424 (X'0978') El descriptor de suscripción (MQSD) no es válido.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') La serie de selección no sigue la sintaxis del selector de IBM MQ y no hay ningún proveedor de selección de mensajes ampliados disponible.

MQRC_SELECTION_STRING_ERROR
2519 (X'09D7') La serie de selección se debe especificar tal como se describe en la documentación de la estructura MQCHARV.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB pero la serie de selección que se ha especificado contenía un error de sintaxis.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') El campo SubUserData no es válido.

MQRC_SUB_NAME_ERROR
2440 (X'0988') El campo SubName no es válido.

MQRC_SUB_ALREADY_EXISTS
2432 (X'0980') La suscripción ya existe.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') El campo SubUserData no es válido.

MQRC_TOPIC_STRING_ERROR
2425 (X'0979') La serie de tema no es válida.

MQRC_UNKNOWN_OBJECT_NAME
2085 (X'0825') No se puede encontrar el objeto identificado en el campo ObjectName de MQSD.

MQRC_SUB_JOIN_NOT_ALTERABLE
29440 (X'7300') La modalidad de uso compartido de suscripciones no es compatible con la suscripción existente. Se puede devolver este error al intentar reanudar una suscripción compartida de JMS 2.0 en una aplicación que no sea JMS.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

- La suscripción se compone de un tema, nombrado utilizando el nombre abreviado de un objeto de tema predefinido, el nombre completo de la serie de tema o formado por la concatenación de dos partes. Consulte la descripción de *ObjectName* y *ObjectString* en "MQSD - Descriptor de suscripción" en la página 572.
- El gestor de colas efectúa comprobaciones de seguridad cuando se emite una llamada MQSUB para verificar que el identificador de usuario con el que se está ejecutando la aplicación tiene el nivel de autorización adecuado antes de permitir el acceso. El objeto de tema adecuado se encuentra en la jerarquía de temas y se realiza una comprobación de autoridad en este objeto de tema para garantizar que se ha establecido la autoridad para suscripción. Si no se utiliza la opción MQSO_MANAGED, se realiza una comprobación de autoridad en la cola de destino para garantizar que se ha establecido la autoridad para salida. Si se utiliza la opción MQSO_MANAGED, no se realiza ninguna comprobación de autoridad en la cola gestionada para acceso de consulta o salida.
- Si no proporciona un Hobj como entrada, la llamada MQSUB asigna dos descriptores de contexto, un descriptor de contexto de objeto (Hobj) y un descriptor de contexto de suscripción (Hsub).
- El Hobj devuelto en la llamada MQSUB cuando se utiliza la opción MQSO_MANAGED se puede consultar para encontrar atributos como por ejemplo el umbral de retroceso o el nombre de recolocación excesiva en cola de retroceso. También puede consultar el nombre de la cola gestionada, pero no debe intentar abrir directamente esta cola.
- Las suscripciones se pueden agrupar, lo que permite que se entregue una única publicación al grupo de suscripciones, incluso si más de una suscripción del grupo coincidía con la publicación. Las suscripciones se pueden agrupar utilizando la opción MQSO_GROUP_SUB, y para agrupar suscripciones estas deben
  - utilizar la misma cola con nombre (es decir, no utilizar la opción MQSO_MANAGED) en el mismo gestor de colas, representado por el parámetro Hobj en la llamada MQSUB
  - compartir el mismo SubCorrelId
  - estar en el mismo SubLevel

  Estos atributos definen el conjunto de suscripciones que se tienen en cuenta para formar parte de un grupo y que también son los atributos que no se pueden alterar si se agrupa una suscripción. Si se altera el SubLevel, se genera MQRC_SUBLEVEL_NOT_ALTERABLE, y si se altera cualquiera de los demás atributos (que se pueden cambiar si una suscripción no está agrupada) se genera MQRC_GROUPING_NOT_ALTERABLE.
- La finalización satisfactoria de la llamada MQSUB no significa que la acción se haya completado. Para comprobar que esta llamada se ha completado, consulte el paso DEFINE SUB en Comprobación de que los mandatos asíncronos en redes distribuidas han finalizado.
- Los campos del MQSD están rellenados cuando se devuelven de una llamada MQSUB que utiliza la opción MQSO_RESUME. El MQSD devuelto se puede pasar directamente en una llamada MQSUB que utiliza la opción MQSO_ALTER con los cambios que es necesario realizar a la suscripción aplicada al MQSD. Algunos campos tienen algunas consideraciones especiales, tal como se indica en la tabla.

Tabla 556. Consideraciones especiales de los campos del MQSD

Nombre de campo en MQSD	Consideraciones especiales
Opciones de acceso o creación	Algunas de las opciones se pueden restablecer cuando se devuelve la llamada MQSUB. Si, a continuación, reutiliza el MQSD en una llamada MQSUB, la opción necesaria se debe establecer explícitamente.
Opciones de durabilidad, Opciones de destino, Opciones de registro & Opciones de comodín	Estas opciones se establecen según corresponda

Tabla 556. Consideraciones especiales de los campos del MQSD (continuación)	
Nombre de campo en MQSD	**Consideraciones especiales**
Opciones de publicación	Estas opciones se establecen según corresponda, excepto MQSO_NEW_PUBLICATIONS_ONLY, que sólo es aplicable a MQSO_CREATE.
Otras opciones	Estas opciones no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
ObjectName	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB.
ObjectString	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB. El nombre de tema completo utilizado en el campo *ResObjectString*, si se proporciona un almacenamiento intermedio.
AlternateUserId y AlternateSecurityId	Estos campos de sólo entrada no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
SubExpiry	En la devolución de una llamada MQSUB mediante la opción MQSO_RESUME, este campo se establece en la caducidad original de la suscripción y no en el tiempo restante para la caducidad. Si a continuación reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER, restablezca la caducidad de la suscripción para reiniciar la cuenta regresiva.
SubName	Este campo es un campo de entrada en una llamada MQSUB y no se modifica en la salida.
SubUserData y SelectionString	Estos campos de longitud variable se devuelven en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio, y también una longitud de almacenamiento intermedio positiva en *VSBufSize*. Si no se proporciona ningún almacenamiento intermedio, sólo se devuelve la longitud en el campo *VSLength* de MQCHARV. Si el almacenamiento intermedio proporcionado es inferior al espacio necesario para devolver el campo, sólo se devuelven *VSBufSize* bytes en el almacenamiento intermedio proporcionado. Si a continuación reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER y no se proporciona ningún almacenamiento intermedio pero se proporciona *VSLength* distinto de cero, si esta longitud coincide con la longitud existente del campo no se realizará ninguna modificación en el campo.

Tabla 556. Consideraciones especiales de los campos del MQSD (continuación)	
Nombre de campo en MQSD	**Consideraciones especiales**
SubCorrelId y PubAccountingToken	Si no utiliza MQSO_SET_CORREL_ID, el gestor de colas genera el *SubCorrelId*. Si no utiliza MQSO_SET_IDENTITY_CONTEXT, el gestor de colas genera *PubAccountingToken*. Estos campos se devuelven en el MQSD de una llamada MQSUB utilizando la opción MQSO_RESUME. Si el gestor de colas los genera, el valor generado se devuelve en una llamada MQSUB utilizando la opción MQSO_CREATE o MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Estos campos se devuelven en el MQSD.
ResObjectString	Este campo de sólo salida se devuelve en el MQSD si se proporciona un almacenamiento intermedio.

Invocación en C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;    /* Connection handle */
MQSD    SubDesc;  /* Subscription descriptor */
MQHOBJ  Hobj;     /* Object handle */
MQHOBJ  Hsub;     /* Subscription handle */
MQLONG  CompCode; /* Completion code */
MQLONG  Reason;   /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Subscription descriptor
 01  SUBDESC.
     COPY CMQSDV.
**   Object handle
 01  HOBJ     PIC S9(9) BINARY.
**   Subscription handle
 01  HSUB     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn    fixed bin(31); /* Connection handle */
dcl SubDesc  like MQSD;     /* Subscription descriptor */
dcl Hobj     fixed bin(31); /* Object handle */
dcl Hsub     fixed bin(31); /* Subscription handle */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason   fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUB,(HCONN,SUBDESC,HOBJ,HSUB,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN    DS      F  Connection handle
SUBDESC  CMQSDA  ,  Subscription descriptor
HOBJ     DS      F  Object handle
HSUB     DS      F  Subscription handle
COMPCODE DS      F  Completion code
REASON   DS      F  Reason code qualifying COMPCODE
```

MQSUBRQ-Solicitud de suscripción

Utilice la llamada MQSUBRQ para realizar una solicitud para la publicación retenida, cuando el suscriptor se haya registrado con MQSO_PUBLICATIONS_ON_REQUEST.

Sintaxis

MQSUBRQ (*Hconn*, *Hsub*, *Acción*, *SubRqOpts*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hsub

Tipo: MQHOBJ - entrada

Este descriptor de contexto representa la suscripción para la que se va a solicitar una actualización. El valor de *Hsub* se ha devuelto de una llamada MQSUB anterior.

Acción

Tipo: MQLONG - entrada

Este parámetro controla la acción concreta que se está solicitando en la suscripción. Se debe especificar el valor siguiente:

MQSR_ACTION_PUBLICATION

Esta acción solicita que se envíe una publicación de actualización para el tema especificado. Sólo se puede utilizar si el suscriptor ha especificado la opción MQSO_PUBLICATIONS_ON_REQUEST en la llamada MQSUB cuando ha realizado la suscripción. Si el gestor de colas tiene una publicación retenida para el tema, se envía al suscriptor. Si no es así, la llamada falla. Si a una

aplicación se le envía una publicación que se ha retenido, esto se indica mediante la propiedad de mensaje MQIsRetained de dicha publicación.

Puesto que el tema de la suscripción existente representada por el parámetro Hsub puede contener comodines, el suscriptor puede recibir varias publicaciones retenidas.

SubRqOpts
Tipo: MQSRO-entrada/salida

Estas opciones controlan la acción de MQSUBRQ, consulte "MQSRO-Opciones de solicitud de suscripción" en la página 595 para obtener más detalles.

Si no se necesitan opciones, los programas escritos en C o S/390 assembler pueden especificar una dirección de parámetro nula en lugar de especificar la dirección de una estructura MQSRO.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón
Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_NO_RETAINED_MSG
2437 (X'0985 ') No hay publicaciones retenidas almacenadas actualmente para este tema.

MQRC_OPTIONS_ERROR
2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR_QUIESCING
2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_SRO_ERROR
2438 (X'0986 ') En la llamada MQSUBRQ, las opciones de solicitud de suscripción MQSRO no son válidas.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Notas de uso

Las siguientes notas de uso se aplican al uso del código de acción MQSR_ACTION_PUBLICATION:

1. Si este verbo se completa correctamente, las publicaciones retenidas que coinciden con la suscripción especificada se han enviado a la suscripción y se pueden recibir utilizando MQGET o MQCB utilizando el Hobj devuelto en el verbo MQSUB original que ha creado la suscripción.
2. Si el tema suscrito por el verbo MQSUB original que ha creado la suscripción contenía un comodín, se puede enviar más de una publicación retenida. El número de publicaciones enviadas como resultado de esta llamada se registra en el campo NumPubs de la estructura SubRqOpts.
3. Si este verbo se completa con un código de razón de MQRC_NO_RETAINED_MSG, no había publicaciones retenidas actualmente para el tema especificado. #
4. Si este verbo se completa con un código de razón de MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, actualmente hay publicaciones retenidas para el tema especificado, pero se ha producido un error que significa que no se han podido entregar.
5. La aplicación debe tener una suscripción actual al tema para poder realizar esta llamada. Si la suscripción se ha realizado en una instancia anterior de la aplicación y no está disponible un descriptor de contexto válido para la suscripción, la aplicación debe llamar primero a MQSUB con la opción MQSO_RESUME para obtener un descriptor de contexto para utilizarla en esta llamada.
6. Las publicaciones se envían al destino que está registrado para su uso con la suscripción actual de esta aplicación. Si las publicaciones deben enviarse a otro lugar, la suscripción debe alterarse primero utilizando la llamada MQSUB con la opción MQSO_ALTER.

Invocación en C

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;     /* Connection handle */
MQHOBJ  Hsub;      /* Subscription handle */
MQLONG  Action;    /* Action requested by MQSUBRQ */
MQSRO   SubRqOpts; /* Options that control the action of MQSUBRQ */
MQLONG  CompCode;  /* Completion code */
MQLONG  Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Action requested by MQSUBRQ
01 ACTION PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
01 SUBRQOPTS.
COPY CMQSROV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn fixed bin(31);    /* Connection handle */
dcl Hsub fixed bin(31);     /* Subscription handle */
dcl Action fixed bin(31);   /* Action requested by MQSUBRQ */
dcl SubRqOpts like MQSRO;   /* Options that control the action of MQSUBRQ */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31);   /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN DS F Connection handle
HSUB DS F Subscription handle
ACTION DS F Action requested by MQSUBRQ
SUBRQOPTS CMQSROA , Options that control the action of MQSUBRQ
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

objeto, atributos de

Esta colección de temas lista sólo los objetos IBM MQ que pueden ser objeto de una llamada a función MQINQ, y proporciona detalles de los atributos que se pueden consultar y los selectores que se van a utilizar.

Atributos para el gestor de colas

Algunos atributos de gestor de colas son fijos para implementaciones concretas; otros se pueden cambiar utilizando el mandato MQSC `ALTER QMGR`.

Los atributos también se pueden visualizar utilizando el mandato `DISPLAY QMGR`. La mayoría de los atributos del gestor de colas se pueden consultar abriendo un objeto MQOT_Q_MGR especial y utilizando la llamada MQINQ con el descriptor de contexto devuelto.

En la tabla siguiente se resumen los atributos que son específicos del gestor de colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con la llamada MQINQ; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más información.

Tabla 557. Atributos para el gestor de colas

Atributo	Descripción
AccountingConnOverride	Alterar temporalmente los valores de contabilidad.
AccountingInterval	Frecuencia con la que se escriben registros de contabilidad intermedios.
ActivityConnOverride	Alterar temporalmente los valores de actividad.
ActivityTrace	Controla la recopilación del rastreo de actividad de la aplicación MQI de IBM MQ .
AdoptNewMCACheck	Elementos seleccionados para determinar si se debe adoptar un nuevo MCA.
AdoptNewMCAType	Indica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado.

Tabla 557. Atributos para el gestor de colas (continuación)	
Atributo	**Descripción**
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
AuthorityEvent	Controla si se generan sucesos de autorización (no autorizados)
BridgeEvent	Atributo de control para sucesos de puente.
ChannelAutoDef	Controla si se permite la definición de canal automática
ChannelAutoDefEvent	Controla si se generan sucesos de definición automática de canal
ChannelAutoDefExit	Nombre de la salida de usuario para la definición automática de canal
ChannelEvent	Atributo de control para sucesos de canal.
ChannelInitiatorControl	Atributo de control para el iniciador de canal
ChannelMonitoring	Datos de supervisión en línea para canales
ChannelStatistics	Controla la recopilación de datos estadísticos para canales.
ChinitAdapters	Número de subtareas de adaptador para procesar llamadas IBM MQ .
ChinitDispatchers	Número de asignadores a utilizar para el iniciador de canal.
	Reservado para uso de IBM .
ChinitTraceAutoStart	Indica si el rastreo de iniciador de canal debe iniciarse automáticamente.
ChinitTraceTableSize	Tamaño del espacio de datos de rastreo del iniciador de canal.
ClusterSenderMonitoringDefault	Datos de supervisión en línea predeterminados para canales emisores de clúster
Estadísticas deClusterSender	Controla la recopilación de información de supervisión de estadísticas para canales emisores de clúster.
ClusterWorkloadData	Datos de usuario para salida de carga de trabajo de clúster
ClusterWorkloadExit	Nombre de salida de usuario para la gestión de carga de trabajo de clúster
ClusterWorkloadLength	Longitud máxima de datos de mensaje pasados a salida de carga de trabajo de clúster
CLWLMRUChannels	Número de canales utilizados más recientemente para el equilibrio de carga de trabajo de clúster
CLWLUseQ	La carga de trabajo de clúster utiliza la cola remota.
CodedCharSetId	Identificador de juego de caracteres codificado
CommandEvent	Atributo de control para sucesos de mandato.
Atributo QName de CommandInput	Nombre de cola de entrada de mandatos
CommandLevel	Nivel de mandato
Atributo Control deCommandServer	Atributo de control para el servidor de mandatos.
Atributo Suceso de configuración	Atributo de control para sucesos de configuración.
DeadLetterQName	Nombre de la cola de mensajes no entregados
DEFCLXQ	Tipo de cola de transmisión de clúster predeterminado
DefXmitQName	Nombre de cola de transmisión predeterminado
DistLists	Soporte de lista de distribución
DNSGroup	Nombre del grupo para el escucha TCP cuando se utiliza el soporte de Workload Manager Dynamic Domain Name Services.
DNSWLM	Indica si el escucha TCP se registra en el Gestor de carga de trabajo para Dynamic Domain Name Services.
ExpiryInterval	Intervalo entre exploraciones de mensajes caducados
IGQPutAuthority	Autorización de transferencia a colas dentro del grupo
IGQUserId	Identificador de usuario de transferencia a colas dentro del grupo
InhibitEvent	Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación)
IPAddressVersion	Versión de la dirección Internet Protocol
IntraGroupqueuing	Soporte de transferencia a colas dentro del grupo
ListenerTimer	Intervalo de tiempo entre los intentos de reiniciar el escucha después de una anomalía de APPC o TCP/IP.

Tabla 557. Atributos para el gestor de colas (continuación)	
Atributo	**Descripción**
LocalEvent	Controla si se generan sucesos de error locales
LoggerEvent	Controla si se generan sucesos de registrador
LUGroupName	Nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas.
LUName	Nombre de LU a utilizar para las transmisiones de LU de salida 6.2 .
LU62ARMSuffix	Sufijo de SYS1.PARMLIB miembro APPCPMxx, que nombra LUADD para este iniciador de canal.
LU62Channels	Número máximo de canales actuales o clientes conectados que utilizan LU 6.2.
MaxActiveChannels	Número máximo de canales que pueden estar activos en cualquier momento.
MaxChannels	Número máximo de canales actuales.
MaxHandles	Número máximo de descriptores de contexto
MaxMsgLength	Longitud máxima de mensaje en bytes
MaxPriority, atributo	Prioridad máxima
MaxPropertiesLength	Longitud máxima de datos de propiedad en bytes
MaxUncommittedMsgs	Número máximo de mensajes no confirmados dentro de una unidad de trabajo
MQIAccounting	Controla la recopilación de información de contabilidad para datos MQI.
MQIStatistics	Controla la recopilación de información de supervisión de estadísticas para el gestor de colas.
MsgMarkBrowseInterval	Intervalo tras el cual el gestor de colas puede eliminar la marca de los mensajes examinados.
OutboundPortMin	Con *OutboundPortMin*, define el rango de números de puerto a utilizar al enlazar canales de salida.
OutboundPortMin	Con *OutboundPortMax*, define el rango de números de puerto a utilizar al enlazar canales de salida.
PerformanceEvent	Controla si se generan sucesos relacionados con el rendimiento
Plataforma	Plataforma en la que se ejecuta el gestor de colas
PubSubNPInputMsg	Si se debe descartar (o conservar) un mensaje de entrada no entregado
PubSubNPResponse	Controla el comportamiento de los no entregados
PubSubMaxMsgRetryCount	Número de reintentos al procesar (bajo punto de sincronismo) un mensaje de mandato fallido
PubSubSyncPoint	Indica si sólo los mensajes persistentes (o todos) deben procesarse bajo punto de sincronismo
PubSubMode	Si la interfaz de publicación/suscripción en cola se está ejecutando
QMgrDesc	Descripción del gestor de colas
QMgrIdentifier	Identificador exclusivo generado internamente del gestor de colas
QMgrName	Nombre del gestor de colas
QSGName	Nombre del grupo de compartición de colas
QueueAccounting	Controla la recopilación de información de contabilidad para colas.
QueueMonitoring	Datos de supervisión en línea para colas
QueueStatistics	Controla la recopilación de datos de estadísticas para colas.
ReceiveTimeout	El tiempo que el canal TCP/IP espera los datos antes de volver al estado inactivo.
ReceiveTimeoutMin	Calificador para *ReceiveTimeout*.
ReceiveTimeoutType	Tiempo mínimo que el canal TCP/IP espera los datos antes de volver al estado inactivo.
RemoteEvent	Controla si se generan sucesos de error remotos
RepositoryName	Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio
RepositoryNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio
ScyCase	Caso de perfiles de seguridad
SharedQMgrNombre	Nombre de gestor de colas compartido
“SPLCAP” en la página 844	IBM MQ Protección avanzada de seguridad de mensajes para un gestor de colas activado o desactivado.
SSLCRLNamelist [1]	Nombre del objeto de lista de nombres que contiene nombres de objetos de información de autenticación.

Tabla 557. Atributos para el gestor de colas (continuación)	
Atributo	**Descripción**
SSLCryptoHardware [1]	Serie de configuración de hardware criptográfico.
SSLEvent	Atributo de control para sucesos TLS.
SSLFIPSRequired	Utilice sólo algoritmos certificados por FIPS para la criptografía.
SSLKeyRepository [1]	Ubicación del repositorio de claves TLS.
Recuento deSSLKeyReset	Recuento de restablecimiento de clave TLS.
SSLTasks [1]	Número de subtareas de servidor para procesar llamadas TLS.
StatisticsInterval	Frecuencia con la que se escriben datos de supervisión de estadísticas.
StartStopEvent	Controla si se generan sucesos de inicio y detención
SyncPoint	Disponibilidad de punto de sincronismo
TCPChannels	Número máximo de canales actuales o clientes conectados que utilizan TCP/IP.
TCPKeepAlive	Indica si se debe utilizar TCP KEEPALIVE para comprobar otro extremo de la conexión.
TCPName	Nombre del sistema TCP/IP que está utilizando.
TCPStackType	Cómo puede utilizar el iniciador de canal las direcciones TCP/IP.
Atributo Registro deTraceRoute	Controla el registro de la información de ruta de rastreo.
TriggerInterval	Desencadenante-intervalo de mensaje
Versión	Versión
XrCapability	Especifica si los mandatos de telemetría están soportados.

Notas:

1. Este atributo no se puede consultar utilizando la llamada MQINQ y no se describe en esta sección. Consulte Cambiar gestor de colas para obtener detalles de este atributo.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

Federal Information Processing Standards (FIPS) para UNIX, Linux, and Windows

Alteración temporal de AccountingConn(MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de los valores ACCTMQI y ACCTQDATA en el atributo Qmgr.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO
: Las aplicaciones no pueden alterar temporalmente el valor de los atributos ACCTMQI y ACCTQ Qmgr utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONNX. Éste es el valor predeterminado.

MQMON_HABILITADO
: Las aplicaciones pueden alterar temporalmente los atributos ACCTQ y ACCTMQI Qmgr utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_CONN_OVERRIDE con la llamada MQINQ.

AccountingInterval (MQLONG)

Especifica cuánto tiempo debe transcurrir antes de que se graben los registros de contabilidad intermedios (en segundos).

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos). Especifique 0 para desactivar los registros intermedios.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Linux
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_INTERVAL con la llamada MQINQ.

Alteración temporal de ActivityConn(MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de ACTVTRC en el atributo del gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO
: Las aplicaciones no pueden alterar temporalmente el valor del atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONNX. Éste es el valor predeterminado.

MQMON_HABILITADO
: Las aplicaciones pueden alterar temporalmente el atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_CONN_OVERRIDE con la llamada MQINQ .

ActivityTrace (MQLONG)

Esto controla la recopilación del rastreo de actividad de la aplicación MQI de IBM MQ .

El valor puede ser uno de los siguientes:

MQMON_ON
: Recopile el rastreo de actividad de la aplicación MQI de IBM MQ .

MQMON_OFF
: No recopile el rastreo de actividad de la aplicación MQI de IBM MQ . Éste es el valor predeterminado.

Si establece el atributo de gestor de colas ACTVCONO en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_TRACE con la llamada MQINQ .

AdoptNewMCACheck (MQLONG)

Define los elementos que se deben comprobar para determinar si se debe adoptar un MCA cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo

El valor puede ser uno de los siguientes:

MQADOPT_CHECK_Q_MGR_NAME
Compruebe el nombre del gestor de colas.

MQADOPT_CHECK_NET_ADDR
Compruebe la dirección de red.

MQADOPT_CHECK_ALL
Compruebe el nombre del gestor de colas y la dirección de red. Si es posible, realice esta comprobación para evitar que los canales se apaguen, de forma involuntaria o maliciosa. Éste es el valor predeterminado.

MQADOPT_CHECK_NONE
No compruebe ningún elemento.

Los cambios en este atributo entrarán en vigor la próxima vez que un canal intente adoptar un canal.

z/OS Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_CHECK con la llamada MQINQ.

AdoptNewMCAType (MQLONG)

Especifica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado cuando se detecta una nueva solicitud de canal de entrada que coincide con el atributo MCACheck AdoptNew

Es uno de los valores siguientes:

MQADOPT_TYPE_NO
La adopción de instancias de canal huérfanas no es necesaria. Éste es el valor predeterminado.

MQADOPT_TYPE_ALL
Adopte todos los tipos de canal.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_TYPE con la llamada MQINQ.

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

Esto controla si se generan sucesos de autorización (no autorizados). Es uno de los valores siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_AUTORITY_EVENT con la llamada MQINQ.

BridgeEvent (MQLONG)

Especifica si se generan sucesos de puente IMS .

El valor puede ser uno de los siguientes:

MQEVR_ENABLED
Genere sucesos de puente IMS , como se indica a continuación:

MQRC_BRIDGE_STARTED
MQRC_BRIDGE_STOPPED

MQEVR_DISABLED
No genere sucesos de puente IMS ; este es el valor predeterminado.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_BRIDGE_EVENT con la llamada MQINQ.

Definición ChannelAuto(MQLONG)

Este atributo controla la definición automática de canales de tipo MQCHT_RECEIVER y MQCHT_SVRCONN. La definición automática de canales MQCHT_CLUSSDR siempre está habilitada. El valor puede ser uno de los siguientes:

MQCHAD_DISABLED
Definición automática de canal inhabilitada.

MQCHAD_ENABLED
Definición automática de canal habilitada.

Multi Este atributo sólo está soportado en Multiplatforms.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF con la llamada MQINQ.

ChannelAutoDefEvent (MQLONG)

Esto controla si se generan sucesos de definición automática de canal. Se aplica a canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Multi Este atributo sólo está soportado en Multiplatforms.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF_EVENT con la llamada MQINQ.

ChannelAutoDefExit (MQCHARn)

Es el nombre de la salida de usuario para la definición automática de canal. Si este nombre no está en blanco y *ChannelAutoDef* tiene el valor MQCHAD_ENABLED, se llama a la salida cada vez que el gestor de colas está a punto de crear una definición de canal. Esto se aplica a los canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. A continuación, la salida puede realizar una de las acciones siguientes:

- Cree la definición de canal sin cambios.
- Modifique los atributos de la definición de canal que se crea.
- Suprimir la creación del canal por completo.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno. Consulte la introducción a la estructura MQCD en "MQCD-Definición de canal" en la página 1517 para obtener detalles sobre el valor de este atributo en diversos entornos.

z/OS En z/OS, este atributo sólo se aplica a los canales de clúster emisor y de clúster receptor.

Para determinar el valor de este atributo, utilice el selector MQCA_CHANNEL_AUTO_DEF_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)

Especifica si se generan sucesos de canal.

Es uno de los valores siguientes:

MQEVR_EXCEPCIÓN
Solo genere los siguientes sucesos de canal:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED con los ReasonQualifierssiguientes:

  MQRQ_CHANNEL_STOPPED_ERROR
  MQRQ_CHANNEL_STOPPED_RETRY
  MQRQ_CHANNEL_STOPPED_DISABLED

  MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED
Generar todos los sucesos de canal. Es decir, además de los generados por EXCEPTION, genera los siguientes sucesos de canal:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED con el ReasonQualifiersiguiente:

  MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED
No genere sucesos de canal; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_EVENT con la llamada MQINQ.

Control de ChannelInitiator(MQLONG)

Especifica si el iniciador de canal debe iniciarse cuando se inicia el gestor de colas.

Es uno de los valores siguientes:

MQSVC_CONTROL_MANUAL
El iniciador de canal no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR
El iniciador de canal se iniciará automáticamente cuando se inicie el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_CONTROL con la llamada MQINQ.

ChannelMonitoring (MQLONG)

Este atributo especifica datos de supervisión en línea para canales.

El valor puede ser uno de los siguientes:

MQMON_NONE
Inhabilite la recopilación de datos para la supervisión de canales para todos los canales independientemente del valor del atributo de canal MONCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de supervisión para los canales que especifiquen QMGR en el atributo de canal MONCHL.

MQMON_LOW
Active la recopilación de datos de supervisión con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_MEDIO
Active la recopilación de datos de supervisión con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_HIGH
Active la recopilación de datos de supervisión con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

z/OS En sistemas z/OS , la habilitación de este parámetro simplemente activa la recopilación de datos de estadísticas, independientemente del valor que seleccione. Si se especifica `LOW`, `MEDIUM` o `HIGH` no hay diferencia en los resultados.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_CHANNEL con la llamada MQINQ.

ChannelStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para los canales.

El valor puede ser uno de los siguientes:

MQMON_NONE
Inhabilite la recopilación de datos para las estadísticas de canal para todos los canales independientemente del valor del atributo de canal STATCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para los canales que especifiquen QMGR en el atributo de canal STATCHL.

MQMON_LOW
Active la recopilación de datos de estadísticas con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_MEDIO
Active la recopilación de datos de estadísticas con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_HIGH
Active la recopilación de datos de estadísticas con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

Para la mayoría de los sistemas, se recomienda utilizar MEDIUM. Sin embargo, para un canal que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

z/OS En sistemas z/OS , la habilitación de este parámetro simplemente activa la recopilación de datos de estadísticas, independientemente del valor que seleccione. Si se especifica `LOW`, `MEDIUM` o `HIGH` no hay diferencia en los resultados.Este parámetro debe estar habilitado para poder recopilar los registros de contabilidad de canal.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_CHANNEL con la llamada MQINQ.

ChinitAdapters (MQLONG)

Es el número de subtareas de adaptador que se deben utilizar para procesar llamadas IBM MQ . El valor debe ser de 0 a 9999, con un valor por omisión de 8.

La proporción de adaptadores con respecto a los asignadores (el atributo ChinitDispatchers) debe ser aproximadamente de 8 a 5. Sin embargo, si sólo tiene pocos canales, no tiene que disminuir el valor de este parámetro del valor predeterminado. Puede utilizar los valores siguientes: para un sistema de prueba, 8 (valor predeterminado); para un sistema de producción, 20. Lo ideal es que tenga 20 adaptadores, lo que proporciona un mayor paralelismo de las llamadas de IBM MQ . Es importante para los mensajes persistentes. Es posible que sea mejor tener menos adaptadores para los mensajes no persistentes.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_ADAPTERS con la llamada MQINQ.

ChinitDispatchers (MQLONG)

Es el número de asignadores que se van a utilizar para el iniciador de canal. El valor debe ser de 0 a 9999, con un valor por omisión de 5.

Como directriz, permita un asignador para 50 canales actuales. Sin embargo, si sólo tiene unos pocos canales, no tiene que disminuir el valor de este atributo del valor predeterminado. Si está utilizando TCP/IP, el mayor número de asignadores que se utilizan para canales TCP/IP es 100, incluso si especifica un valor mayor aquí. Puede utilizar los valores siguientes: sistemas de prueba, 5 (el valor predeterminado); sistemas de producción, 20 (necesita 20 asignadores para manejar hasta 1000 canales activos).

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_DISPATCHERS con la llamada MQINQ.

ChinitTraceAutoStart (MQLONG)

Especifica si se debe iniciar automáticamente el rastreo de iniciador de canal.

El valor puede ser uno de los siguientes:

MQTRAXSTR_SÍ
Iniciar automáticamente el rastreo del iniciador de canal. Éste es el valor predeterminado.

MQTRAXSTR_NO
No inicie automáticamente el rastreo del iniciador de canal.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_AUTO_START con la llamada MQINQ.

ChinitTraceTableSize (MQLONG)

Es el tamaño del espacio de datos de rastreo del iniciador de canal (en MB).

El valor debe estar en el rango de 0 a 2048, con un valor predeterminado de 2.

Nota: Siempre que utilice espacios de datos de z/OS grandes, asegúrese de que tiene suficiente almacenamiento auxiliar en el sistema para dar soporte a cualquier actividad de paginación de z/OS relacionada. También tendrá que aumentar el tamaño de los conjuntos de datos SYS1.DUMP.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_TABLE_SIZE con la llamada MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Especifica el valor que debe sustituirse por el atributo ChannelMonitoring de los canales emisores de clúster definidos automáticamente.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
La recopilación de datos de supervisión en línea se hereda del valor del atributo **`ChannelMonitoring`** del gestor de colas. Este es el valor predeterminado.

MQMON_OFF
La supervisión del canal está inhabilitada

MQMON_LOW
A menos que *`ChannelMonitoring`* sea MQMON_NONE, la supervisión se habilita con una tasa baja de recopilación de datos con un efecto mínimo en el rendimiento del sistema. No es probable que los datos recopilados sean los más actuales.

MQMON_MEDIO
A menos que *`ChannelMonitoring`* sea MQMON_NONE, la supervisión se habilita con una tasa moderada de recopilación de datos con un efecto limitado en el rendimiento del sistema.

MQMON_HIGH
A menos que *`ChannelMonitoring`* sea MQMON_NONE, la supervisión se habilita con una alta tasa de recopilación de datos con un efecto probable en el rendimiento del sistema. Los datos recopilados son los más actuales disponibles.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_AUTO_CLUSSDR con la llamada MQINQ.

Estadísticas de ClusterSender(MQLONG)

Puesto que los canales emisores de clúster se pueden definir automáticamente a partir de la definición de CLUSRCVR en el repositorio, no puede modificar el valor del atributo STATCHL para estos canales emisores de clúster definidos automáticamente utilizando el canal ALTER. Para estos canales, la decisión de recopilar datos de supervisión en línea se basa en el valor de este atributo de gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
La recopilación de datos de estadísticas para canales emisores de clúster definidos automáticamente se basa en el valor del atributo de gestor de colas STATCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente.

MQMON_LOW
Habilite la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción baja de recopilación de datos.

MQMON_MEDIO
Habilite la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción moderada de recopilación de datos.

MQMON_HIGH
Habilite la recopilación de datos de estadísticas para los canales de clúster emisor definidos automáticamente con una proporción alta de recopilación de datos.

Para la mayoría de los sistemas recomendamos MEDIO. Sin embargo, para un canal emisor de clúster definido automáticamente que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

z/OS En sistemas z/OS , la habilitación de este parámetro simplemente activa la recopilación de datos de estadísticas, independientemente del valor que seleccione. Si se especifica `LOW`, `MEDIUM` o `HIGH` no hay diferencia en los resultados.Este parámetro debe estar habilitado para poder recopilar los registros de contabilidad de canal.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_AUTO_CLUSSDR con la llamada MQINQ.

Datos de ClusterWorkload(MQCHAR32)

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_DATA con la llamada MQINQ.

Salida de ClusterWorkload(MQCHARn)

Es el nombre de la salida de usuario para la gestión de carga de trabajo de clúster. Si este nombre no está en blanco, se llama a la salida cada vez que se transfiere un mensaje a una cola de clúster o se mueve de una cola de clúster emisor a otra. A continuación, la salida puede aceptar la instancia de cola seleccionada por el gestor de colas como destino del mensaje o seleccionar otra instancia de cola.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

Longitud de ClusterWorkload(MQLONG)

Es la longitud máxima de los datos de mensaje que se pasan a la salida de carga de trabajo del clúster. La longitud real de los datos pasados a la salida es el mínimo de lo siguiente:

- Longitud del mensaje.
- El atributo **`MaxMsgLength`** del gestor de colas.
- El atributo **`ClusterWorkloadLength`** .

Para determinar el valor de este atributo, utilice el selector MQIA_CLUSTER_WORKLOAD_LENGTH con la llamada MQINQ.

CLWLMRUChannels (MQLONG)

Especifica el número máximo de canales de clúster utilizados más recientemente, que se deben tener en cuenta para que los utilice el algoritmo de elección de carga de trabajo de clúster.

Es un valor comprendido entre 1 y 999999999.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_MRU_CHANNELS con la llamada MQINQ.

CLWLUseQ (MQLONG)

Especifica si se deben utilizar colas remotas para la carga de trabajo del clúster.

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY
: Utilice colas locales y remotas.

MQCLWL_USEQ_LOCAL
: No utilice colas remotas. Este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_USEQ con la llamada MQINQ.

CodedCharSetId (MQLONG)

Define el juego de caracteres utilizado por el gestor de colas para todos los campos de serie de caracteres definidos en la MQI como, por ejemplo, los nombres de los objetos y la fecha y hora de creación de la cola. El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de objeto. No se aplica a los datos de aplicación transportados en el mensaje. El valor depende del entorno:

- En z/OS, el valor se establece a partir de los parámetros del sistema cuando se inicia el gestor de colas; el valor predeterminado es 500.
- En Windows, el valor es el CODEPAGE primario del usuario que crea el gestor de colas.
- En IBM i, el valor es el que se establece en el entorno cuando se crea por primera vez el gestor de colas.
- En UNIX, el valor es el valor predeterminado CODESET para el entorno local del usuario que crea el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CODED_CHAR_SET_ID con la llamada MQINQ.

CommandEvent (MQLONG)

Especifica si se generan sucesos de mandato, como se indica a continuación:

MQEVR_DISABLED
: No generar sucesos de mandato. Éste es el valor predeterminado.

MQEVR_ENABLED
: Generar sucesos de mandato.

MQEVR_NO_DISPLAY
: Los sucesos de mandato se generan para todos los mandatos satisfactorios distintos de MQINQ.

Para determinar el valor de este atributo, utilice el selector MQIA_COMMAND_EVENT con la llamada MQINQ.

CommandInputQName (MQCHAR48)

Es el nombre de la cola de entrada de mandatos definida en el gestor de colas local. Es una cola a la que los usuarios pueden enviar mandatos, si están autorizados a hacerlo. El nombre de la cola depende del entorno:

- En z/OS, el nombre de la cola es SYSTEM.COMMAND.INPUT; se le pueden enviar los mandatos MQSC y PCF. Consulte Mandatos MQSC para obtener detalles de los mandatos MQSC y Definiciones de los formatos de mandatos programables para obtener detalles de los mandatos PCF.
- En todos los demás entornos, el nombre de la cola es SYSTEM.ADMIN.COMMAND.QUEUEy solo se le pueden enviar mandatos PCF. Sin embargo, se puede enviar un mandato MQSC a esta cola si el mandato MQSC está entre un mandato PCF de tipo MQCMD_ESCAPE. Consulte Escape para obtener información sobre el mandato Escape.

Para determinar el valor de este atributo, utilice el selector MQCA_COMMAND_INPUT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Nota: V 9.1.0 Se ha eliminado el soporte para el sistema operativo HP-UX para todos los componentes de IBM MQ , incluidos el servidor y los clientes.

Indica el nivel de mandatos de control del sistema soportados por el gestor de colas. Puede tener uno de los valores siguientes:

MQCMDL_LEVEL_710
Nivel 710 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM WebSphere MQ for AIX 7.1
- IBM WebSphere MQ for HP-UX 7.1
- IBM WebSphere MQ for IBM i 7.1
- IBM WebSphere MQ for Linux 7.1
- IBM WebSphere MQ for Solaris 7.1
- IBM WebSphere MQ for Windows 7.1
- IBM WebSphere MQ for z/OS 7.1

MQCMDL_LEVEL_750
Nivel 750 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM WebSphere MQ for AIX 7.5
- IBM WebSphere MQ for HP-UX 7.5
- IBM WebSphere MQ for IBM i 7.5
- IBM WebSphere MQ for Linux 7.5
- IBM MQ for Solaris 7.5
- IBM WebSphere MQ for Windows 7.5

MQCMDL_LEVEL_800
Nivel 800 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM MQ for AIX 8.0
- IBM MQ for HP-UX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0
- IBM MQ for Solaris 8.0
- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

MQCMDL_LEVEL_801
Nivel 801 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
- IBM MQ for Solaris 8.0.0 Fix Pack 2

MQCMDL_LEVEL_802
Nivel 802 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for HP-UX 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 Solaris 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3

MQCMDL_LEVEL_900
Nivel 900 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM MQ for AIX 9.0
- IBM MQ for HP-UX 9.0
- IBM MQ for IBM i 9.0
- IBM MQ for Linux 9.0
- IBM MQ for Solaris 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

MQCMDL_LEVEL_901
Nivel 901 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 902 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 903 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 904 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 905 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 910 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- IBM MQ for AIX 9.1.0
- IBM MQ for IBM i 9.1.0
- IBM MQ for Linux 9.1.0
- IBM MQ for Solaris 9.1.0
- IBM MQ for Windows 9.1.0
- IBM MQ for z/OS 9.1.0

MQCMDL_LEVEL_911
Nivel 911 de los mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 912 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 913 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 914 de los mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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
Nivel 915 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes:

- 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

El conjunto de mandatos de control del sistema que corresponde a un valor determinado del atributo **CommandLevel** varía según el valor del atributo **Platform** ; ambos deben utilizarse para decidir qué mandatos de control del sistema están soportados.

Para determinar el valor de este atributo, utilice el selector `MQIA_COMMAND_LEVEL` con la llamada `MQINQ` .

CommandServerControl (MQLONG)

Especifica si el servidor de mandatos debe iniciarse cuando se inicia el gestor de colas.

El valor puede ser cualquiera de los valores siguientes:

MQSVC_CONTROL_MANUAL
El servidor de mandatos no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR
El servidor de mandatos se iniciará automáticamente cuando se inicie el gestor de colas.

Este atributo no está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_CMD_SERVER_CONTROL con la llamada MQINQ.

ConfigurationEvent (MQLONG)

Controla si se generan sucesos de configuración.

Para determinar el valor de este atributo, utilice el selector MQIA_CONFIGURATION_EVENT con la llamada MQINQ.

El valor puede ser cualquiera de los valores siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

V 9.1.5 Multi *Tamaño de CurrentQFile(MQLONG)*

El tamaño actual del archivo de cola en megabytes, redondeado al megabyte más cercano.

Tabla 558. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

El valor de este atributo de estado de cola es el tamaño que tenga actualmente la cola, redondeado al megabyte más cercano. Para una cola nueva con atributos predeterminados, el valor de **CurrentQFileSize** es 1.

El valor máximo de este atributo es 99,999,9999 MB y no hay ningún valor predeterminado para este atributo.

V 9.1.5 Multi *CurrentMaxQFileSize (MQLONG)*

El tamaño máximo actual al que puede crecer el archivo de cola, redondeado al megabyte más cercano, dado el tamaño de bloque actual en uso en una cola.

Tabla 559. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

El uso de este campo es dos veces:

- Si establece **MaxQFileSize** en el valor predeterminado para el tamaño de bloque actual, **CurrentMaxQFileSize** muestra el valor real al que equivale el valor predeterminado.
- Si **CurrentMaxQFileSize** no coincide con **MaxQFileSize**, sabe que la cola debe drenarse para poder adoptar una granularidad mayor.

Nota: Consulte Modificación de archivos de cola de IBM MQ para obtener más información sobre cómo cambiar el tamaño de los archivos de cola y el tamaño de bloque y la granularidad.

El valor máximo de este atributo es 99,999,9999 MB y no hay ningún valor predeterminado. El valor es cualquiera que sea el valor máximo establecido actualmente; para una cola nueva con los atributos predeterminados, el valor de **CurrentMaxQFileSize** es 2.088.960 MB.

DeadLetterQName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local como la cola de mensajes no entregados (undelivered-message). Los mensajes se envían a esta cola si no pueden direccionarse a su destino correcto.

Por ejemplo, los mensajes se colocan en esta cola cuando:

- Llega un mensaje a un gestor de colas, destinado a una cola que todavía no está definida en ese gestor de colas
- Un mensaje llega a un gestor de colas, pero la cola a la que está destinado no puede recibirlo porque, posiblemente:
  - La cola está llena
  - Las solicitudes de colocación están inhibidas
  - El nodo emisor no tiene autorización para colocar mensajes en la cola

Las aplicaciones también pueden colocar mensajes en la cola de mensajes no entregados.

Los mensajes de informe se tratan de la misma forma que los mensajes ordinarios; si el mensaje de informe no se puede entregar a su cola de destino (normalmente la cola especificada por el campo *ReplyToQ* en el descriptor de mensaje del mensaje original), el mensaje de informe se coloca en la cola de mensajes no entregados (mensaje no entregado).

Nota: Mensajes que han pasado su tiempo de caducidad (consulte MQMD-Campo de caducidad) **no** se transfieren a esta cola cuando se descartan. Sin embargo, todavía se genera un mensaje de informe de caducidad (MQRO_EXPIRATION) y se envía a la cola *ReplyToQ* , si lo solicita la aplicación emisora.

Los mensajes no se colocan en la cola de mensajes no entregados cuando la aplicación que ha emitido la solicitud de colocación ha recibido una notificación síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 (por ejemplo, un mensaje colocado en una cola local para el que se inhiben las solicitudes de colocación).

Los mensajes de la cola de mensajes no entregados a veces tienen sus datos de mensaje de aplicación con el prefijo de una estructura MQDLH. Esta estructura contiene información adicional que indica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message). Consulte "MQDLH - Cabecera de mensajes no entregados" en la página 349 para obtener más detalles de esta estructura.

Esta cola debe ser una cola local, con un atributo **Usage** de MQUS_NORMAL.

Si un gestor de colas no da soporte a una cola de mensajes no entregados (undelivered-message), o no se ha definido una, el nombre estará en blanco. Todos los gestores de colas de IBM MQ dan soporte a una cola de mensajes no entregados (undelivered-message), pero de forma predeterminada no está definida.

Si la cola de mensajes no entregados (undelivered-message) no está definida, llena o inutilizable por alguna otra razón, un mensaje que un agente de canal de mensajes le habría transferido se conserva en la cola de transmisión.

Para determinar el valor de este atributo, utilice el selector MQCA_DEAD_LETTER_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

DefClusterXmitQueueTipo (MQLONG)

El atributo DefClusterXmitQueueTipo controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Los valores de **DefClusterXmitQueueType** son MQCLXQ_SCTQ o MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Todos los canales de clúster emisor envían mensajes de SYSTEM.CLUSTER.TRANSMIT.QUEUE. El correlID de los mensajes colocados en la cola de transmisión identifica el canal de clúster emisor al que va destinado el mensaje.

SCTQ se establece cuando se define un gestor de colas. Este comportamiento es implícito en las versiones de IBM WebSphere MQ anteriores a la IBM WebSphere MQ 7.5. En versiones anteriores, el atributo de gestor de colas DefClusterXmitQueueType no estaba presente.

MQCLXQ_CHANNEL

Cada canal de clúster emisor envía mensajes desde una cola de transmisión diferente. Cada cola de transmisión se crea como una cola dinámica permanente de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Si el atributo de gestor de colas, DefClusterXmitQueueTipo, se establece en CHANNEL, la configuración predeterminada cambia a los canales de clúster emisor que se están asociando con colas de transmisión de clúster individuales. Las colas de transmisión son colas dinámicas permanentes creadas a partir de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Cada cola de transmisión está asociada a un canal de clúster emisor. Cuando un canal de clúster emisor presta servicio a una cola de transmisión de clúster, la cola de transmisión contiene mensajes únicamente para un gestor de colas de un clúster. Puede configurar clústeres de modo que cada gestor de colas de un clúster sólo contenga una cola de clúster. En este caso, el tráfico de mensajes de un gestor de colas a cada cola de clúster se transfiere por separado de los mensajes a otras colas.

Para consultar el valor, llame a MQINQo envíe un mandato Consultar gestor de colas (MQCMD_INQUIRE_Q_MGR) PCF, estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Para cambiar el valor, envíe un mandato PCF Cambiar gestor de colas (MQCMD_CHANGE_Q_MGR), estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Referencia relacionada

Cambiar gestor de colas

Consultar gestor de colas

"MQINQ-Consultar atributos de objeto" en la página 715

La llamada `MQINQ` devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

DefXmitQName (MQCHAR48)

Es el nombre de la cola de transmisión que se utiliza para la transmisión de mensajes a gestores de colas remotos, si no hay ninguna otra indicación de qué cola de transmisión utilizar.

Si no hay ninguna cola de transmisión predeterminada, el nombre está totalmente en blanco. El valor inicial de este atributo está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_DEF_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Esto indica si el gestor de colas local da soporte a listas de distribución en las llamadas MQPUT y MQPUT1 . Es uno de los valores siguientes:

MQDL_SUPPORTED
Listas de distribución soportadas.

MQDL_NOT_SUPPORTED
Listas de distribución no soportadas.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ.

DNSGroup (MQCHAR18)

Este parámetro ya no se utiliza. Consulte Qué ha cambiado en IBM MQ 8.0.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_DNS_GROUP con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (MQLONG)

Este parámetro ya no se utiliza. Consulte Qué ha cambiado en IBM MQ 8.0.

El valor puede ser uno de los siguientes:

MQDNSWLM_SÍ
Este valor se puede ver en un gestor de colas migrado desde un release anterior. El valor se ignora.

MQDNSWLM_NO
Este es el único valor soportado por el gestor de colas.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_DNS_WLM con la llamada MQINQ.

ExpiryInterval (MQLONG)

Esto indica la frecuencia con la que el gestor de colas explora las colas en busca de mensajes caducados. Es un intervalo de tiempo en segundos comprendido entre 1 y 99 999 999, o el siguiente valor especial:

MQEXPI_OFF
El gestor de colas no explora las colas en busca de mensajes caducados.

Para determinar el valor de este atributo, utilice el selector MQIA_EXPIRY_INTERVAL con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

IGQPutAuthority (MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartición de colas. Indica el tipo de comprobación de autorización que se realiza cuando el agente de transferencia a colas dentro del grupo local (agente IGQ) elimina un mensaje de la cola de transmisión compartida y coloca el mensaje en una cola local. El valor puede ser uno de los siguientes:

MQIGQPA_PREDETERMINADO
El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el identificador de usuario del agente de IGQ local (*IGQUserId*).

CONTEXTO_MQIGQPA
El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueban el identificador de usuario del agente de IGQ local (*IGQUserId*) y el valor del campo *UserIdentifier* en el MQMD *incorporado* . Este último identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

MQIGQPA_ONLY_IGQ
El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, este identificador de usuario se utiliza para todas las comprobaciones.

MQIGQPA_ALTERNATE_OR_IGQ
El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el valor del campo *UserIdentifier* en el MQMD *incorporado* . Este identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

Para determinar el valor de este atributo, utilice el selector MQIA_IGQ_PUT_AUTHORITY con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

IGQUserId (MQLONG)

Este atributo sólo es aplicable si el gestor de colas local es miembro de un grupo de compartición de colas. Especifica el identificador de usuario que está asociado con el agente de transferencia a colas dentro del grupo local (agente IGQ). Este identificador es uno de los identificadores de usuario que se pueden comprobar para la autorización cuando el agente de IGQ coloca mensajes en colas locales. Los identificadores de usuario reales que se comprueban dependen del valor del atributo **IGQPutAuthority** y de las opciones de seguridad externas.

Si *IGQUserId* está en blanco, no se asocia ningún identificador de usuario con el agente de IGQ y no se realiza la comprobación de autorización correspondiente (aunque es posible que se sigan comprobando otros identificadores de usuario para la autorización).

Para determinar el valor de este atributo, utilice el selector MQCA_IGQ_USER_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_USER_ID_LENGTH.

z/OS Este atributo sólo está soportado en z/OS.

InhibitEvent (MQLONG)

Esto controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación). El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

IntraGroupqueuing (MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartición de colas. Indica si la transferencia a colas dentro del grupo está habilitada para el grupo de compartición de colas. El valor puede ser uno de los siguientes:

MQIGQ_INHABILITADO
Todos los mensajes destinados a otros gestores de colas del grupo de compartición de colas se transmiten utilizando canales convencionales.

MQIGQ_HABILITADO
Los mensajes destinados a otros gestores de colas del grupo de compartición de colas se transmiten utilizando la cola de transmisión compartida si se cumple la condición siguiente:

- La longitud de los datos del mensaje más la cabecera de transmisión no supera los 63 KB (64 512 bytes).

  Se recomienda asignar algo más de espacio que el tamaño de MQXQH para la cabecera de transmisión; para ello se proporciona la constante MQ_MSG_HEADER_LENGTH.

Si esta condición no se satisface, el mensaje se transmite utilizando canales convencionales.

Nota: Cuando se habilita la transferencia a colas dentro del grupo, el orden de los mensajes transmitidos utilizando la cola de transmisión compartida no se conserva en relación con los transmitidos utilizando canales convencionales.

Para determinar el valor de este atributo, utilice el selector MQIA_INTRA_GROUP_queuing con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

IPAddressVersion (MQLONG)

Especifica qué versión de dirección IP, ya sea IPv4 o IPv6, se utiliza.

Este atributo sólo es relevante para los sistemas que ejecutan IPv4 y IPv6 y sólo afecta a los canales definidos como que tienen un *TransportType* de MQXPY_TCP cuando se cumple una de las condiciones siguientes:

- El *ConnectionName* del canal es un nombre de host que se resuelve en una dirección IPv4 y IPv6 y no se especifica su parámetro **LocalAddress** .
- Los *ConnectionName* y *LocalAddress* del canal son ambos nombres de host que se resuelven en las direcciones IPv4 y IPv6 .

El valor puede ser cualquiera de los valores siguientes:

MQIPADDR_IPv4
IPv4 .

MQIPADDR_IPv6
IPv6 .

Para determinar el valor de este atributo, utilice el selector MQIA_IP_ADDRESS_VERSION con la llamada MQINQ.

ListenerTimer (MQLONG)

Es el intervalo de tiempo (en segundos) entre los intentos de IBM MQ de reiniciar el escucha si se ha producido una anomalía de APPC o TCP/IP. El valor debe estar entre 5 y 9999, con un valor predeterminado de 60.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_LISTENER_TIMER con la llamada MQINQ.

LocalEvent (MQLONG)

Esto controla si se generan sucesos de error locales. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_LOCAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

LoggerEvent (MQLONG)

Esto controla si se generan sucesos de registro de recuperación. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_LOGGER_EVENT con la llamada MQINQ.

Multi Este atributo sólo está soportado en Multiplatforms.

LUGroupName (MQCHAR8)

Es el nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas. Si deja este nombre en blanco, no puede utilizar este escucha.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU_GROUP_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_LU_NAME_LENGTH.

LUName (MQCHAR8)

Es el nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 . Establézcalo en la misma LU que utiliza el escucha para las transmisiones de entrada. Si deja este nombre en blanco, se

utiliza la LU predeterminada APPC/MVS; es variable, por lo tanto, establezca siempre LUName si utiliza LU6.2.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Es el sufijo de SYS1.PARMLIB APPCPMxx, que nomina el LUADD para este iniciador de canal. El mandato z/OS SET APPC=xx se emite cuando ARM reinicia el iniciador de canal. Si deja este nombre en blanco, no se emite SET APPC=xx.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU62_ARM_SUFFIX con la llamada MQINQ. La longitud de este atributo la proporciona MQ_ARM_SUFFIX_LENGTH.

LU62Channels (MQLONG)

Es el número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 .

El valor debe estar en el rango de 0 a 9999, con un valor predeterminado de 200. Si lo establece en cero, no se utiliza el protocolo de transmisión LU 6.2 .

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_LU62_CHANNELS con la llamada MQINQ.

Canales MaxActive(MQLONG)

Este atributo es el número máximo de canales que pueden estar *activos* en cualquier momento.

El valor predeterminado es el especificado en el atributo `MaxChannels`.

Para z/OS, el valor debe estar en el rango de 1 a 9 999.

Para todas las demás plataformas, el valor predeterminado es 999.999.999, lo que significa que el número de canales activos es ilimitado, o puede establecerse en un número real para imponer un límite.

El parámetro **`MaxActiveChannels`** es un atributo de gestor de colas sólo en z/OS . En las otras plataformas, **`MaxActiveChannels`** es un atributo del archivo `qm.ini` . Consulte Stanzas del archivo de configuración para colas distribuidas para obtener información sobre cómo establecer el atributo **`MaxActiveChannels`** en otras plataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVE_CHANNELS con la llamada **`MQINQ`** .

Conceptos relacionados

Estados de un canal

MaxChannels (MQLONG)

Este atributo es el número máximo de canales que pueden ser *actuales* (incluidos los canales de conexión de servidor con clientes conectados).

Para z/OS, el valor debe estar en el rango de 1 a 9 999, con un valor predeterminado de 200.

Un sistema que está ocupado sirviendo conexiones desde la red puede necesitar un número mayor que el valor predeterminado. Determine el valor que es correcto para su entorno, idealmente observando el comportamiento del sistema durante las pruebas.

En todas las demás plataformas, el valor predeterminado es 100. Puede establecer **`MaxChannels`** en un valor distinto para limitar el número máximo del canales actuales si es necesario.

El parámetro **MaxChannels** es un atributo de gestor de colas sólo en z/OS . En las otras plataformas, **MaxChannels** es un atributo del archivo `qm.ini` . Consulte Stanzas del archivo de configuración para colas distribuidas para obtener información sobre cómo establecer el atributo **MaxChannels** en otras plataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_CHANNELS con la llamada **MQINQ** .

Conceptos relacionados

Estados de un canal

MaxHandles (MQLONG)

Es el número máximo de descriptores de contexto abiertos que cualquier tarea puede utilizar simultáneamente. Cada llamada MQOPEN satisfactoria para una sola cola (o para un objeto que no es una cola) utiliza un descriptor de contexto. Ese descriptor de contexto pasa a estar disponible para su reutilización cuando se cierra el objeto. Sin embargo, cuando se abre una lista de distribución, a cada cola de la lista de distribución se le asigna un descriptor de contexto independiente, y de modo que la llamada MQOPEN utiliza tantos descriptores de contexto como colas hay en la lista de distribución. Esto debe tenerse en cuenta al decidir un valor adecuado para *MaxHandles*.

La llamada MQPUT1 realiza una llamada MQOPEN como parte de su proceso; como resultado, MQPUT1 utiliza tantos manejadores como MQOPEN lo haría, pero los manejadores sólo se utilizan durante la propia llamada MQPUT1 .

En z/OS, *tarea* significa una tarea CICS , una tarea MVS o una región dependiente de IMS .

El valor está en el rango de 1 a 999.999.999. El valor predeterminado lo determina el entorno:

- En z/OS, el valor predeterminado es 100.
- En todos los demás entornos, el valor predeterminado es 256.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MANEJAR con la llamada MQINQ.

MaxMsgLongitud (MQLONG)

Es la longitud del mensaje *físico* más largo que el gestor de colas puede manejar. Sin embargo, debido a que el atributo de gestor de colas **MaxMsgLength** se puede establecer independientemente del atributo de cola **MaxMsgLength** , el mensaje físico más largo que se puede colocar en una cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, una aplicación puede colocar un mensaje lógico que sea más largo que el menor de los dos atributos **MaxMsgLength** , pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

El límite inferior para el atributo **MaxMsgLength** es de 32 KB (32.768 bytes). El límite superior es de 100 MB (104 857 600 bytes).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxPriority (MQLONG)

Esta es la prioridad de mensaje máxima soportada por el gestor de colas. Las prioridades van de cero (más bajo) a *MaxPriority* (más alto).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PRIORITY con la llamada MQINQ.

MaxPropertiesLongitud (MQLONG)

Se utiliza para controlar el tamaño de las propiedades que pueden fluir con un mensaje. Esto incluye tanto el nombre de propiedad en bytes como el tamaño del valor de propiedad también en bytes.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PROPERTIES_LENGTH con la llamada MQINQ.

V 9.1.5 Multi *Tamaño de MaxQFile(MQLONG)*

Tamaño máximo, en megabytes, hasta el que puede crecer un archivo de cola.

Tabla 560. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es posible que un archivo de cola supere el tamaño máximo, si está configurado en un valor inferior al tamaño de archivo de cola actual. Si esto sucede, el archivo de cola ya no acepta nuevos mensajes, pero permite que se consuman los mensajes existentes. Cuando el tamaño del archivo de cola ha caído por debajo del valor configurado, se permite transferir mensajes nuevos a la cola.

Nota: Esta figura puede diferir del valor del atributo configurado en la cola, porque internamente el gestor de colas puede necesitar utilizar un tamaño de bloque mayor para alcanzar el tamaño elegido. Consulte Modificación de archivos de cola de IBM MQ para obtener más información sobre cómo cambiar el tamaño de los archivos de cola y el tamaño de bloque y la granularidad.

Cuando la granularidad necesita cambiar porque este atributo se ha aumentado, el mensaje de aviso AMQ7493W `Granularidad cambiada` se graba en los registros de AMQERR. Esto le proporciona una indicación de que necesita planificar que la cola se vacíe, para que IBM MQ adopte la nueva granularidad.

El valor máximo de este atributo es 267.386.880 MB y el valor predeterminado, y el valor migrado, es 2.088.960 MB, que es el máximo actual para una cola con una granularidad igual a 512.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_Q_FILE_SIZE con la llamada MQINQ.

Mensajes MaxUncommitted(MQLONG)

Es el número máximo de mensajes no confirmados que pueden existir en una unidad de trabajo. El número de mensajes no confirmados es la suma de los siguientes mensajes desde el inicio de la unidad de trabajo actual:

- Mensajes transferidos por la aplicación con la opción MQPMO_SYNCPOINT
- Mensajes recuperados por la aplicación con la opción MQGMO_SYNCPOINT
- Mensajes de activación y mensajes de informe COA generados por el gestor de colas para mensajes transferidos con la opción MQPMO_SYNCPOINT
- Mensajes de informe COD generados por el gestor de colas para mensajes recuperados con la opción MQGMO_SYNCPOINT

Los mensajes siguientes no se cuentan como no confirmados:

- Mensajes colocados o recuperados por la aplicación fuera de una unidad de trabajo
- Mensajes desencadenantes o mensajes de informe COA/COD generados por el gestor de colas como resultado de mensajes colocados o recuperados fuera de una unidad de trabajo
- Mensajes de informe de caducidad generados por el gestor de colas (incluso si la llamada que provoca el mensaje de informe de caducidad ha especificado MQGMO_SYNCPOINT)
- Mensajes de suceso generados por el gestor de colas (incluso si la llamada que provoca el mensaje de suceso ha especificado MQPMO_SYNCPOINT o MQGMO_SYNCPOINT)

Nota:

1. Los mensajes de informe de excepción los genera el agente de canal de mensajes (MCA), o la aplicación, y se tratan de la misma forma que los mensajes ordinarios colocados o recuperados por la aplicación.

2. Cuando se coloca un mensaje o segmento con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en uno independientemente de cuántos mensajes físicos resulten realmente de la colocación. (Puede producirse más de un mensaje físico si el gestor de colas debe subdividir el mensaje o segmento.)
3. Cuando se coloca una lista de distribución con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en un *para cada mensaje físico que se genera*. Esto puede ser tan pequeño como uno, o tan grande como el número de destinos en la lista de distribución.

El límite inferior para este atributo es 1; el límite superior es 999 999 999. El valor predeterminado es 10000.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_UNCOMMITTED_MSGS con la llamada MQINQ.

MQIAccounting (MQLONG)

Esto controla la recopilación de información de contabilidad para datos MQI.

El valor puede ser uno de los siguientes:

MQMON_ON
Recopilar datos de contabilidad de API.

MQMON_OFF
No recopile datos de contabilidad de API. Éste es el valor predeterminado.

Si establece el atributo de gestor de colas ACCTCONO en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO. Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_MQI con la llamada MQINQ.

MQIStatistics (MQLONG)

Esto controla la recopilación de información de supervisión de estadísticas para el gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_ON
Recopilar estadísticas de MQI.

MQMON_OFF
No recopile estadísticas MQI. Éste es el valor predeterminado.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_MQI con la llamada MQINQ.

MsgMarkBrowseInterval (MQLONG)

Intervalo de tiempo en milisegundos después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Se trata de un intervalo de tiempo (en milisegundos) después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Este atributo describe el intervalo de tiempo durante el cual se espera que los mensajes que se han marcado como examinados por una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_CO_OP, permanezcan marcados como examinados.

El gestor de colas puede desmarcar automáticamente los mensajes examinados que se han marcado como examinados para el conjunto de descriptores de contexto cooperantes cuando se han marcado durante más de este intervalo aproximado.

Esto no afecta al estado de ningún mensaje marcado como examinar, que se ha obtenido mediante una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_HANDLE.

El valor máximo es 999.999.999 y el valor predeterminado es 5000. Un valor especial de -1 para `MsgMarkBrowseInterval` representa un intervalo de tiempo ilimitado.

Atención: Este valor no debe estar por debajo del valor predeterminado de 5000.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_MARK_BROWSE_INTERVAL con la llamada MQINQ.

OutboundPortMáx (MQLONG)

Es el número de puerto más alto del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se utilizarán para enlazar canales de salida.

El valor es un entero en el rango de 0 a 65535, y debe ser igual o mayor que el valor mínimo de OutboundPort. El valor por omisión es 0.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MAX con la llamada MQINQ.

OutboundPortmínimo (MQLONG)

Es el número de puerto más bajo del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se van a utilizar para enlazar canales de salida.

El valor es un entero en el rango de 0 a 65535, y debe ser igual o menor que el valor máximo de OutboundPort. El valor por omisión es 0.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MIN con la llamada MQINQ.

PerformanceEvent (MQLONG)

Esto controla si se generan sucesos relacionados con el rendimiento. Es uno de los valores siguientes:

MQEVR_DISABLED
: Informes de sucesos inhabilitados.

MQEVR_ENABLED
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_PERFORMANCE_EVENT con la llamada MQINQ.

Plataforma (MQLONG)

Esto indica el sistema operativo en el que se ejecuta el gestor de colas:

MQPL_AIX
AIX (mismo valor que MQPL_UNIX).

DISPOSITIVO_MQPL_APPLIANCE
IBM MQ Appliance

MQPL_MVS
z/OS (mismo valor que MQPL_ZOS).

MQPL_OS390
z/OS (mismo valor que MQPL_ZOS).

MQPL_OS400
IBM i.

MQPL_UNIX
UNIX.

MQPL_WINDOWS_NT
Windows .

MQPL_ZOS
z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_PLATFORM con la llamada MQINQ.

PubSubNPInputMsg (MQLONG)

Indica si se debe descartar o mantener un mensaje de entrada no entregado.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_DISCARD
Los mensajes de entrada no persistentes se pueden desechar si no se pueden procesar.

Este es el valor predeterminado.

MQUNDELIVERED_KEEP
Los mensajes de entrada no persistentes no se desecharán si no se pueden procesar. En esta situación, la interfaz de publicación/suscripción en cola continuará reintentando el proceso a intervalos adecuados y no continúa procesando mensajes posteriores.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_MSG con la llamada MQINQ.

PubSubNPResponse (MQLONG)

Controla el comportamiento de los mensajes de respuesta no entregados.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_NORMAL
Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se colocan en la cola de mensajes no entregados, si no se pueden colocar en la DLQ, se descartan.

MQUNDELIVERED_SAFE
Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se transfieren a la cola de mensajes no entregados. Si la respuesta no se puede establecer y no se puede colocar en la DLQ, la interfaz de publicación/suscripción en cola retrotraerá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados y no continuará procesando los mensajes posteriores.

MQUNDELIVERED_DISCARD
Las respuestas no persistentes no se colocan en la cola de respuestas se descartan.

Este es el valor predeterminado para los nuevos gestores de colas.

MQUNDELIVERED_KEEP
Las respuestas no persistentes no se colocan en la cola de mensajes no entregados ni se descartan. En su lugar, la interfaz de publicación/suscripción en cola restituirá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_RESP con la llamada MQINQ.

Valor predeterminado para los gestores de colas migrados.

Si el gestor de colas se ha migrado desde IBM MQ V6.0, el valor inicial de este atributo depende de los valores de *DiscardNonPersistentResponse* y *DLQNonPersistentResponse* antes de la migración, tal como se muestra en la tabla siguiente.

		Respuesta DLQNonPersistent		
		Sí	**No**	**No establecido**
DiscardNonPersistentResponse	**Sí**	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD	MQUNDELIVERED_NORMAL
	No	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	No establecido	Si SyncPointPersistent = No, MQUNDELIVERED_SAFE de lo contrario MQUNDELIVERED_NORMAL	Si SyncPointPersistent = No, MQUNDELIVERED_KEEP de lo contrario MQUNDELIVERED_DISCARD	Si SyncPointPersistent = No, MQUNDELIVERED_SAFE de lo contrario MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

Número de reintentos al procesar un mensaje de mandato fallido bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

0 - 999 999 999
El valor predeterminado es 5.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MAXMSG_RETRY_COUNT con la llamada MQINQ.

PubSubSyncPoint (MQLONG)

Indica si sólo los mensajes persistentes o todos los mensajes se procesan bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

MQSYNCPOINT_IFPER
Esto hace que la interfaz de publicación/suscripción en cola reciba mensajes no persistentes fuera del punto de sincronización. Si el daemon recibe una publicación fuera del punto de sincronismo, el daemon reenvía la publicación a los suscriptores que conoce fuera del punto de sincronismo.

Este es el valor predeterminado.

MQSYNCPOINT_YES
Esto hace que la interfaz de publicación/suscripción en cola reciba todos los mensajes bajo punto de sincronismo.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_SYNC_PT con la llamada MQINQ.

Modalidad PubSub(MQLONG)

Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando, permitiendo por lo tanto que las aplicaciones publiquen/suscriban utilizando la interfaz de programación de aplicaciones y las colas que están siendo supervisadas por la interfaz de publicación/suscripción en cola.

El valor puede ser uno de los siguientes:

MQPSM_COMPAT
El motor de publicación/suscripción está ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. La interfaz de publicación/suscripción en cola no se está ejecutando, por lo tanto, no se actúa sobre ningún mensaje que se coloque en las

colas supervisadas por la interfaz de publicación/suscripción en cola. Este valor se utiliza para la compatibilidad con WebSphere Message Broker V6 o versiones anteriores que utilizan este gestor de colas, porque debe leer las mismas colas de las que normalmente lee la interfaz de publicación/suscripción en cola.

MQPSM_INHABILITADO
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola no están ejecutándose. Por lo tanto, no es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. No se actúa sobre los mensajes de publicación/suscripción que se colocan en las colas supervisadas por la interfaz de publicación/suscripción en cola.

MQPSM_HABILITADO
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones y las colas supervisadas por la interfaz de publicación/suscripción en cola. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MODE con la llamada MQINQ.

QMgrDesc (MQCHAR64)

Utilice este campo para un comentario que describa el gestor de colas. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **`CodedCharSetId`**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

- En z/OS, el valor predeterminado es el nombre de producto y el número de versión.
- En todos los demás entornos, el valor por omisión son espacios en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_DESC_LENGTH.

QMgrIdentifier (MQCHAR48)

Es un nombre exclusivo generado internamente para el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_IDENTIFIER con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_IDENTIFIER_LENGTH.

Este atributo está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

y clientes de IBM MQ conectados a estos sistemas.

QMgrName (MQCHAR48)

Es el nombre del gestor de colas local, es decir, el nombre del gestor de colas al que está conectada la aplicación.

Los primeros 12 caracteres del nombre se utilizan para construir un identificador de mensaje exclusivo (consulte MQMD-campo MsgId). Por lo tanto, los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para que los identificadores de mensajes sean exclusivos en la red de gestores de colas.

En z/OS, el nombre es el mismo que el nombre del subsistema, que está limitado a 4 caracteres no en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

QSGName (MQCHAR4)

Es el nombre del grupo de compartición de colas al que pertenece el gestor de colas local. Si el gestor de colas local no pertenece a un grupo de compartición de colas, el nombre está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_QSG_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_QSG_NAME_LENGTH.

z/OS Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Esto controla la recopilación de información de contabilidad para las colas.

El valor puede ser uno de los siguientes:

MQMON_NONE
No recopile datos de contabilidad para colas, independientemente del valor del atributo de contabilidad de cola ACCTQ. Este es el valor predeterminado.

MQMON_OFF
No recopile datos de contabilidad para colas que especifiquen QMGR en el atributo de cola ACCTQ.

MQMON_ON
Recopilar datos de contabilidad para colas que especifican QMGR en el atributo de cola ACCTQ.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Especifica el valor predeterminado para la supervisión en línea de colas.

Si el atributo de cola **QueueMonitoring** se establece en MQMON_Q_MGR, este atributo especifica el valor que asume el canal. El valor puede ser:

MQMON_OFF
La recopilación de datos de supervisión en línea está desactivada. Este es el valor predeterminado inicial del gestor de colas.

MQMON_NONE
La recopilación de datos de supervisión en línea está desactivada para las colas independientemente del valor de su atributo **QueueMonitoring** .

MQMON_LOW
La recopilación de datos de supervisión en línea está activada, con una proporción baja de recopilación de datos.

MQMON_MEDIO
La recopilación de datos de supervisión en línea está activada, con una proporción moderada de recopilación de datos.

MQMON_HIGH
La recopilación de datos de supervisión en línea está activada, con una proporción alta de recopilación de datos.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para las colas.

Es uno de los valores siguientes:

MQMON_NONE
No recopile estadísticas de cola para colas, independientemente del valor del atributo de cola **QueueStatistics** . Este es el valor predeterminado.

MQMON_OFF
No recopile datos de estadísticas para colas que especifiquen el gestor de colas en el atributo de cola **QueueStatistics** .

MQMON_ON
Recopilar datos estadísticos para las colas que especifican el gestor de colas en el atributo de cola **QueueStatistics** .

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_Q con la llamada MQINQ.

ReceiveTimeout (MQLONG)

Especifica cuánto tiempo espera un canal TCP/IP para recibir datos, incluyendo pulsaciones, de su asociado antes de volver al estado inactivo. Sólo se aplica a los canales de mensajes y no a los canales MQI.

El significado exacto de ReceiveTimeout se modifica por el valor especificado en el tipo ReceiveTimeout. El tipo ReceiveTimeoutse puede establecer en uno de los siguientes:

- MQRCVTIME_EQUAL-este valor es el número en segundos que el canal debe esperar. Especifique un valor en el rango de 0 a 999999.
- MQRCVTIME_ADD-este valor es el número en segundos que se debe añadir al HBINT negociado y determina cuánto tiempo espera un canal. Especifique un valor en el rango de 1 a 999999.
- MQRCVTIME_MULTIPLY-este valor es un multiplicador que se aplica al HBINT negociado. Especifique un valor de 0 o un valor en el rango de 2 a 99.

El valor por omisión es 0.

Establezca el tipo ReceiveTimeouten MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0, para impedir que un canal exceda el tiempo de espera para recibir datos de su socio.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT con la llamada MQINQ.

ReceiveTimeoutMín (MQLONG)

Es el tiempo mínimo, en segundos, que un canal TCP/IP espera para recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo.

Sólo se aplica a los canales de mensajes, no a los canales MQI. El valor debe estar en el rango de 0 a 999999, con un valor por omisión de 0.

Si utiliza el tipo ReceiveTimeoutpara especificar que el tiempo de espera del canal TCP/IP debe calcularse en relación con el valor negociado de HBINT, y el valor resultante es menor que el valor de este parámetro, este valor se utiliza en su lugar.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_MIN con la llamada MQINQ.

Tipo ReceiveTimeout(MQLONG)

Este es el calificador, aplicado a ReceiveTimeout para definir cuánto tiempo espera un canal TCP/IP para recibir datos, incluidas las pulsaciones, de su socio, antes de volver al estado inactivo. Sólo se aplica a los canales de mensajes, no a los canales MQI.

El valor puede ser uno de los siguientes:

MQRCVTIME_MULTIPLY
: ReceiveTimeout es un multiplicador que se aplica al valor de HBINT negociado para determinar cuánto tiempo espera un canal. Éste es el valor predeterminado.

MQRCVTIME_ADD
: ReceiveTimeout es un valor, en segundos, que se añade al valor de HBINT negociado para determinar cuánto tiempo espera un canal.

MQRCVTIME_EQUAL
: ReceiveTimeout es un valor, en segundos, que el canal espera.

Para detener un canal que excede el tiempo de espera para recibir datos de su socio, establezca el tipo ReceiveTimeouten MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_TYPE con la llamada MQINQ.

RemoteEvent (MQLONG)

Esto controla si se generan sucesos de error remotos. Es uno de los valores siguientes:

MQEVR_DISABLED
: Informes de sucesos inhabilitados.

MQEVR_ENABLED
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_REMOTE_EVENT con la llamada MQINQ.

RepositoryName (MQCHAR48)

Es el nombre de un clúster para el que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio para más de un clúster, *RepositoryNamelist* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *RepositoryName* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RepositoryNamelist (MQCHAR48)

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio sólo para un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *RepositoryName* para especificar el nombre del clúster, en cuyo caso *RepositoryNamelist* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Especifica si el gestor de colas da soporte a nombres de perfil de seguridad en mayúsculas y minúsculas, o sólo en mayúsculas.

El valor puede ser uno de los siguientes:

MQSCYC_UPPER
Los nombres de perfil de seguridad deben estar en mayúsculas.

MQSCYC_MIXED
Los nombres de perfil de seguridad pueden estar en mayúsculas o en mayúsculas y minúsculas.

Los cambios en este atributo entran en vigor cuando se ejecuta un mandato Renovar seguridad con *SecurityType(MQSECTYPE_CLASSES)* especificado.

z/OS Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SECURITY_CASE con la llamada MQINQ.

SharedQMgrNombre (MQLONG)

Especifica si *ObjectQmgrName* debe utilizarse o tratarse como el gestor de colas local en una llamada MQOPEN, para una cola compartida, cuando *ObjectQmgrName* es el de otro gestor de colas del grupo de compartición de colas.

El valor puede ser cualquiera de los valores siguientes:

MQSQQM_USE
Se utiliza *ObjectQmgrName* y se abre la cola de transmisión adecuada.

MQSQQM_IGNORE
Si la cola de destino es compartida, y *ObjectQmgrName* es la de un gestor de colas en el mismo grupo de compartición de colas, la apertura se realiza localmente.

Este atributo sólo es válido en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SHARED_Q_Q_MGR_NAME con la llamada MQINQ.

SPLCAP

Indica si las prestaciones de seguridad de Advanced Message Security están disponibles para un gestor de colas.

MQCAP_SUPPORTED
Este es el valor predeterminado si el componente AMS está instalado para la instalación bajo la que se ejecuta el gestor de colas.

MQCAP_NO_SOPORTADO

SSLEvent (MQLONG)

Especifica si se generan sucesos TLS.

Es uno de los valores siguientes:

MQEVR_ENABLED
Genere sucesos TLS, como se indica a continuación:

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED
No generar sucesos TLS; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_EVENT con la llamada MQINQ.

SSLFIPSRequired (MQLONG)

Esto le permite especificar que sólo se utilizarán algoritmos certificados por FIPS si la criptografía se ejecuta en IBM MQ, en lugar de en hardware de cifrado. Si el hardware de cifrado está configurado, los módulos de cifrado utilizados son los módulos proporcionados por el producto de hardware; estos módulos pueden o no estar certificados por FIPS a un nivel determinado en función del producto de hardware en uso.

El valor es uno de los valores siguientes:

MQSSL_FIPS_NO
Utilice cualquier CipherSpec soportada en la plataforma en uso. Este es el valor predeterminado.

MQSSL_FIPS_YES
Utilice sólo algoritmos criptográficos certificados por FIPS en las CipherSpecs permitidas en todas las conexiones TLS desde y hacia este gestor de colas.

Este parámetro sólo es válido en plataformas UNIX, Linux, Windowsy z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_FIPS_REQUIRED con la llamada MQINQ.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

Federal Information Processing Standards (FIPS) para UNIX, Linux, and Windows

Recuento de SSLKeyReset(MQLONG)

Especifica cuándo los agentes de canal de mensajes (MCA) de canal TLS que inician la comunicación restablecen la clave secreta utilizada para el cifrado en el canal.

El valor representa el número total de bytes no cifrados que se envían y se reciben a través del canal antes de renegociar la clave secreta. El número de bytes incluye la información de control que envía el MCA.

El valor es un número comprendido entre 0 y 999 999 999, con un valor por omisión de 0. Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. Esto es para evitar el coste de proceso de restablecimientos de clave excesivos que se producirían para valores pequeños de restablecimiento de clave secreta TLS.

La clave secreta se renegocia cuando el número total de bytes no cifrados enviados y recibidos por el MCA del canal iniciador excede el valor especificado. Si las pulsaciones de canal están habilitadas, la clave secreta se renegocia antes de que se envíen o reciban los datos después de una pulsación de canal, o cuando el número total de bytes no cifrados excede el valor especificado, lo que ocurra primero.

El recuento de bytes enviados y recibidos para renegociación incluye la información de control enviada y recibida por el canal MCA y se restablece cuando se produce una renegociación.

Utilice un valor de 0 para indicar que las claves secretas nunca se renegocian.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_RESET_COUNT con la llamada MQINQ.

Suceso StartStop(MQLONG)

Esto controla si se generan sucesos de inicio y detención. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_START_STOP_EVENT con la llamada MQINQ.

StatisticsInterval (MQLONG)

Especifica la frecuencia (en segundos) con la que se graban los datos de supervisión de estadísticas en la cola de supervisión.

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos).

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_INTERVAL con la llamada MQINQ.

SyncPoint (MQLONG)

Esto indica si el gestor de colas local soporta unidades de trabajo y sincronización con las llamadas MQGET, MQPUT y MQPUT1 .

MQSP_AVAILABLE
Unidades de trabajo y sincronización disponibles.

MQSP_NOT_AVAILABLE
Unidades de trabajo y sincronización no disponibles.

- En z/OS este valor nunca se devuelve.

Para determinar el valor de este atributo, utilice el selector MQIA_SYNCPOINT con la llamada MQINQ.

TCPChannels (MQLONG)

Es el número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP.

El valor debe estar en el rango de 0 a 9999, con un valor predeterminado de 200. Si especifica 0, no se utiliza TCP/IP.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_CHANNELS con la llamada MQINQ.

TCPKeepAlive (MQLONG)

Especifica si se debe utilizar TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal se cierra.

El valor puede ser uno de los siguientes:

MQTCPKEEP_YES
Utilice TCP KEEPALIVE tal como se especifica en el conjunto de datos de configuración de perfil TCP. Si especifica el atributo de canal KeepAliveInterval (KAINT), se utiliza el valor en el que se establece.

MQTCPKEEP_NO
No utilice TCP KEEPALIVE. Éste es el valor predeterminado.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_KEEP_ALIVE con la llamada MQINQ.

TCPName (MQCHAR8)

Es el nombre de la pila TCP/IP única o preferida que se utilizará, en función del valor de TCPStackType. Este parámetro sólo es aplicable en varios entornos de pila CINET. El valor predeterminado es TCPIP.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_TCP_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TCP_NAME_LENGTH.

TCPStackType (MQLONG)

Especifica si el iniciador de canal sólo puede utilizar la pila TCP/IP especificada en TCPName, o puede enlazarse opcionalmente con cualquier pila TCP/IP seleccionada. Este parámetro sólo es aplicable en varios entornos de pila CINET.

El valor puede ser uno de los siguientes:

MQTCPSTACK_SINGLE
El iniciador de canal sólo puede utilizar los espacios de direcciones TCP/IP especificados en TCPName. Éste es el valor predeterminado.

MQTCPSTACK_MULTIPLE
El iniciador de canal puede utilizar cualquier espacio de direcciones TCP/IP disponible para él. El valor predeterminado es el especificado en TCPName si no se especifica ningún otro para un canal o escucha.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_STACK_TYPE con la llamada MQINQ.

Registro de TraceRoute(MQLONG)

Esto controla el registro de la información de ruta de rastreo.

El valor puede ser uno de los siguientes:

MQRECORDING_INHABILITADO
No se permite añadir mensajes de ruta de rastreo.

MQRECORDING_Q
Coloque los mensajes de ruta de rastreo en una cola con nombre fijo.

MQRECORDING_MSG
Transfiera los mensajes de ruta de rastreo a una cola determinada utilizando el propio mensaje. Este es el valor predeterminado

Para determinar el valor de este atributo, utilice el selector MQIA_TRACE_ROUTE_REGISTRAR con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte Desencadenamiento de canales.

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola

no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte Desencadenamiento de canales.

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

Versión (MQCFST)

Esta es la versión del código IBM MQ como VVRRMMFF, donde:

VV-Versión

RR-Release

MM-Nivel de mantenimiento

FF-Nivel de arreglo

XrCapability(MQLONG)

Esto controla si los mandatos MQ Telemetry están soportados por el gestor de colas.

El valor puede ser uno de los siguientes:

MQCAP_SUPPORTED
Se da soporte al componente de MQ Telemetry instalado y a los mandatos de telemetría.

MQCAP_NO_SOPORTADO
El componente MQ Telemetry no está instalado.

Este atributo sólo está soportado en las plataformas siguientes:

- IBM i
- UNIX
- Windows

Para determinar el valor de este atributo, utilice el selector MQIA_XR_ABILITY con la llamada MQINQ .

Atributos para colas

Hay cinco tipos de definición de cola. Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola.

Tipos de cola

El gestor de colas da soporte a los siguientes tipos de definición de cola:

Cola local
Puede almacenar mensajes en una cola local.

z/OS En z/OS puede convertirlo en una cola compartida o privada.

Una cola se conoce en un programa como *local* si es propiedad del gestor de colas al que está conectado el programa. Puede obtener mensajes y transferirlos en las colas locales.

El objeto de definición de cola contiene la información de definición de la cola, así como los mensajes físicos colocados en cola.

Cola del gestor de colas local
La cola existe en el gestor de colas local.

z/OS La cola se conoce como cola privada en z/OS.

z/OS **Cola compartida (sóloz/OS)**
La cola existe en un repositorio compartido al que pueden acceder todos los gestores de colas que pertenecen al grupo de compartición de colas propietario del repositorio compartido.

Las aplicaciones conectadas a cualquier gestor de colas del grupo de compartición de colas pueden colocar mensajes y eliminar mensajes de colas de este tipo. Estas colas son efectivamente las mismas que las colas locales. El valor del atributo de cola **QType** es `MQQT_LOCAL`.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminar mensajes de colas de este tipo. El valor del atributo de cola **QType** es `MQQT_LOCAL`.

Cola de clúster

Puede almacenar mensajes en una cola de clúster en el gestor de colas donde está definido. Una cola de clúster es una cola que se aloja en un gestor de colas de clúster y que está disponible para otros gestores de colas del clúster. El valor del atributo de cola **QType** es `MQQT_CLUSTER`.

Una definición de cola de clúster se anuncia en otros gestores de colas del clúster. Los otros gestores de colas del clúster pueden transferir mensajes a una cola de clúster sin necesidad de que haya una definición de cola remota correspondiente. Una cola de clúster se puede anunciar en más de un clúster utilizando una lista de nombres de clúster.

Cuando se anuncia una cola, cualquier gestor de colas del clúster puede poner mensajes en ella. Para transferir un mensaje, el gestor de colas debe averiguar, en los repositorios completos, donde está alojada la cola. A continuación, añade información de direccionamiento al mensaje y pone el mensaje a una cola de transmisión de clúster.

Un gestor de colas puede almacenar mensajes para otros gestores de colas en un clúster en varias colas de transmisión. Puede configurar un gestor de colas para almacenar mensajes en varias colas de transmisión de clúster de dos maneras diferentes. Si establece el atributo de gestor de colas **DEFCLXQ** en `CHANNEL`, se crea automáticamente una cola de transmisión de clúster diferente de `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE` para cada canal de clúster emisor. Si establece la opción de cola de transmisión `CLCHNAME` para que coincida con uno o varios canales de clúster emisor, el gestor de colas puede almacenar mensajes para los canales coincidentes en esa cola de transmisión.

Atención: Si utiliza `SYSTEM.CLUSTER.TRANSMIT.QUEUES` dedicado con un gestor de colas que se ha actualizado desde una versión del producto anterior a IBM WebSphere MQ 7.5, asegúrese de que `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE` tiene la opción SHARE/NOSHARE establecida en **SHARE**.

z/OS Una cola de clúster puede ser una cola que se comparte entre miembros de un grupo de compartición de colas en IBM MQ for z/OS.

Cola remota
Una cola remota no es una cola física; es la definición local de una cola que existe en un gestor de colas remoto. La definición local de la cola remota contiene información que indica al gestor de colas local cómo direccionar los mensajes al gestor de colas remoto.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola de transmisión local que se utiliza para direccionar los mensajes al gestor de colas remoto. Las aplicaciones no pueden eliminar mensajes de colas remotas. El valor del atributo de cola **QType** es `MQQT_REMOTE`.

También puede utilizar una definición de cola remota para:

- Alias de cola de respuestas

  En este caso, el nombre de la definición es el nombre de una cola de respuestas. Para obtener más información, consulte Alias de cola de respuesta y clústeres.
- Alias de gestor de colas

En este caso, el nombre de la definición es un alias para un gestor de colas, no el nombre de una cola. Para obtener más información, consulte Alias y clústeres de gestores de colas.

Cola alias
No es una cola física; es un nombre alternativo para una cola local, una cola compartida, una cola de clúster o una cola remota. El nombre de la cola en la que se resuelve el alias forma parte de la definición de la cola alias.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola en la que se resuelve el alias. Las aplicaciones pueden eliminar mensajes de colas de este tipo si el alias se resuelve en una cola local, una cola compartida o una cola de clúster que tiene una instancia local. El valor del atributo de cola **QType** es `MQQT_ALIAS`.

Cola modelo
No es una cola física; es un conjunto de atributos de cola a partir de los cuales se puede crear una cola local.

Los mensajes no se pueden almacenar en colas de este tipo.

límites en cola

V 9.1.0.5

A partir de la IBM MQ 9.1.0 Fix Pack 5, de forma predeterminada el gestor de colas limita el tamaño máximo de archivo de cola en 2 TB.

Atributos de colas

Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola. Los tipos de cola a los que se aplica un atributo se muestran en las tablas Tabla 561 en la página 850 y posteriores.

La Tabla 561 en la página 850 resume los atributos específicos de las colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con las llamadas `MQINQ` y `MQSET` ; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más detalles.

En la tabla siguiente, las columnas se aplican de la forma siguiente:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si se consulta cualquier otro atributo, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

  Si la cola de clúster se abre solo para consultas, o para consultas y salidas, además de especificar el nombre del gestor de colas base, en su lugar se aplica la columna para colas locales.

Tabla 561. Atributos para colas

Atributo	Descripción	Local	Modelo	Alias	Remotos	Clúster
`AlterationDate`	Fecha en que se modificó por última vez la definición	X		X	X	
`AlterationTime`	Hora a la que se modificó por última vez la definición	X		X	X	

Tabla 561. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
BackoutRequeueQName	Nombre de cola de reposición en cola de restitución excesivo	X	X			
BackoutThreshold	Umbral de restituciones	X	X			
BaseQName	Nombre de cola al que se resuelve el alias			X		
CFStrucName	Nombre de estructura de recurso de acoplamiento	X	X			
CLCHNAME	Nombres de canal de clúster emisor	✓	✓			
ClusterName	Nombre del clúster al que pertenece la cola	X		X	X	X
ClusterNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola	X		X	X	
CLWLQueuePriority	Prioridad de cola de carga de trabajo de clúster	X		X	X	X
CLWLQueueRank	Rango de cola de carga de trabajo de clúster	X		X	X	X
CLWLUseQ	Utilizar cola remota	X				
CreationDate	Fecha de creación de la cola	X				
CreationTime	Hora a la que se ha creado la cola	X				
CurrentQDepth	Profundidad de cola actual	X				
DefaultPutResponse	Resp predet de transferencia	✓	✓	✓	✓	
DefBind	Enlace predeterminado	X		X	X	X
DefinitionType attribute	Tipo de definición de cola	X	X			
DefInputOpenOption	Opción abierta de entrada predeterminada	X	X			
DefPersistence	Persistencia de mensajes predeterminada	X	X	X	X	X
DefPriority	Prioridad de mensajes predeterminada	✓	✓	✓	✓	✓
DefReadAhead	Lectura anticipada predeterminada	X	X	X		
DistLists	Soporte de lista de distribución	X	X			
HardenGetBackout	Si se debe mantener un recuento de restituciones preciso	X	X			
IndexType	Tipo de índice	X	X			
InhibitGet	Si se permiten operaciones get para la cola	X	X	X		
InhibitPut	Si se permiten operaciones de colocación para la cola	X	X	X	X	X
InitiationQName	Nombre de cola de inicio	X	X			
MaxMsgLength	Longitud máxima de mensaje en bytes	X	X			
MaxQDepth	Profundidad máxima de la cola	X	X			
MsgDeliverySequence attribute	Secuencia de entrega de mensajes	X	X			

Tabla 561. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
NonPersistentMessage Class	Objetivo de fiabilidad para mensajes no persistentes	X	X			
OpenInputCount	Número de aperturas para entrada	X				
OpenOutputCount	Número de aperturas para salida	X				
PropertyControl	Control de propiedad	✓	✓	✓		
ProcessName	Nombre de proceso	X	X			
QDepthHighEvent attribute	Si se generan sucesos de profundidad de cola alta	X	X			
QDepthHighLimit	Límite alto para profundidad de cola	X	X			
QDepthLowEvent attribute	Si se generan sucesos de profundidad de cola baja	X	X			
QDepthLowLimit attribute	Límite bajo para profundidad de cola	X	X			
QDepthMaxEvent	Si se generan sucesos de cola llena	X	X			
QDesc	Descripción de la cola	X	X	X	X	X
QName	Nombre de cola	X		X	X	X
QServiceInterval	Destino para intervalo de servicio de cola	X	X			
QServiceIntervalEvent attribute	Si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto	X	X			
QSGDisp attribute	Disposición de grupo de uso compartido de colas	X		X	X	
QueueAccounting	Recopilación de datos de contabilidad de cola	X	X	X	X	X
QueueMonitoring	Datos de supervisión en línea para colas	X	✓			
QueueStatistics	recopilación de datos de estadísticas de cola	X	X	X	X	X
QType	Tipo de cola	X		X	X	X
RemoteQMgrName	Nombre del gestor de colas remoto				X	
RemoteQName	Nombre de cola remota				X	
RetentionInterval	Intervalo de retención	X	X			
Scope	Si también existe una entrada para la cola en un directorio de célula	X		X	X	
Shareability	Compartibilidad de cola	X	X			
StorageClass	Clase de almacenamiento para cola	X	X			
TriggerControl	Activar control	X	X			
TriggerData	Datos desencadenantes	X	X			
TriggerDepth	Profundidad de desencadenante	X	X			
TriggerMsgPriority	Prioridad de mensaje de umbral para desencadenantes	X	X			
TriggerType	Tipo de desencadenante	X	X			
Usage attribute	Uso de la cola	X	X			

Tabla 561. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
XmitQName	Nombre de cola de transmisión				X	

Conceptos relacionados

Colas de clúster

Colas locales

AlterationDate (MQCHAR12)

Fecha en la que se cambió por última vez la definición.

Tabla 562. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes (por ejemplo, `1992-09-23¬¬`, donde ¬¬ representa dos caracteres en blanco).

Los valores de determinados atributos (por ejemplo, *`CurrentQDepth`*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *`AlterationDate`*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Hora a la que se modificó por última vez la definición.

Tabla 563. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS` utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, `09.10.20`).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.

Los valores de determinados atributos (por ejemplo, *`CurrentQDepth`*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *`AlterationTime`*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

BackoutRequeueQName (MQCHAR48)

Este es el nombre de cola de reposición en cola de restitución excesivo. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Tabla 564. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar dónde deben ir los mensajes que se han restituido. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

IBM MQ classes for JMS utiliza este atributo para determinar dónde transferir un mensaje que ya se ha restituido el número máximo de veces especificado por el atributo *BackoutThreshold* .

Para determinar el valor de este atributo, utilice el selector MQCA_BACKOUT_REQ_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)

Este es el umbral de restitución. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Tabla 565. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizarán este atributo para determinar si se debe restituir un mensaje. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

IBM MQ classes for JMS utiliza este atributo para determinar cuántas veces se debe permitir que se restituya un mensaje antes de transferir el mensaje a la cola especificada por el atributo *BackoutRequeueQName* .

Para determinar el valor de este atributo, utilice el selector MQIA_BACKOUT_THRESHOLD con la llamada MQINQ.

BaseQName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local.

Tabla 566. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
		X		

(Para obtener más información sobre los nombres de cola, consulte MQOD-Campo ObjectName.) La cola es de uno de los tipos siguientes:

MQQT_LOCAL
Cola local.

MQQT_REMOTE
Definición local de una cola remota.

MQQT_CLUSTER
Cola de clúster.

Para determinar el valor de este atributo, utilice el selector MQCA_BASE_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

BaseType (MQCFIN)

El tipo de objeto en el que se resuelve el alias.

Tabla 567. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
		X		

Es uno de los valores siguientes:

MQOT_Q
El tipo de objeto base es una cola

MQOT_TOPIC
El tipo de objeto base es un tema

CFStrucName (MQCHAR12)

Es el nombre de la estructura del recurso de acoplamiento donde se almacenan los mensajes de la cola. El primer carácter del nombre está en el rango de A a Z, y los caracteres restantes están en el rango de A a Z, de 0 a 9 o en blanco.

Tabla 568. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Para obtener el nombre completo de la estructura en el recurso de acoplamiento, sufijo el valor del atributo del gestor de colas **QSGName** con el valor del atributo de cola **CFStrucName** .

Este atributo sólo se aplica a las colas compartidas; se ignora si *QSGDisp* no tiene el valor MQQSGD_SHARED.

Para determinar el valor de este atributo, utilice el selector MQCA_CF_STRUC_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CF_STRUC_NAME_LENGTH.

z/OS Este atributo sólo está soportado en z/OS.

ClusterChannelNombre (MQCHAR20)

`ClusterChannelNombre` es el nombre genérico de los canales de clúster emisor que utilizan esta cola como cola de transmisión. El atributo especifica los canales de clúster emisor han enviado mensajes a un canal de clúster receptor desde esta cola de transmisión de clúster.

Tabla 569. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

La configuración del gestor de colas predeterminado es para todos los canales de clúster emisor para enviar mensajes desde una sola cola de transmisión, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. La configuración predeterminada se puede cambiar modificando el atributo del gestor de colas, **DefClusterXmitQueueType**. El valor predeterminado del atributo es `SCTQ`. Puede cambiar el valor a `CHANNEL`. Si establece el atributo **DefClusterXmitQueueType** en `CHANNEL`, cada canal de clúster emisor utiliza de forma predeterminada una cola de transmisión de clúster específica, `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`*.

También puede establecer el atributo de cola de transmisión `ClusterChannelName` en un canal de clúster emisor manualmente. Los mensajes destinados al gestor de colas conectado por el canal de clúster emisor se almacenan en la cola de transmisión que identifica el canal de clúster emisor.

No se almacenan en la cola de transmisión de clúster predeterminada. Si establece el atributo `ClusterChannelName` en blancos, el canal conmuta a la cola de transmisión de clúster predeterminada cuando se reinicia el canal. La cola predeterminada es `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`* o `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, en función del valor del atributo `DefClusterXmitQueueType` del gestor de colas.

Al especificar asteriscos, "`*`", en **`ClusterChannelName`**, puede asociar una cola de transmisión con un conjunto de canales de clúster emisor. Los asteriscos pueden estar al principio, al final o en cualquier posición intermedia de la serie de nombre de canal. **`ClusterChannelName`** está limitado a una longitud de 20 caracteres: `MQ_CHANNEL_NAME_LENGTH`.

ClusterName (MQCHAR48)

Es el nombre del clúster al que pertenece la cola.

Tabla 570. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Si la cola pertenece a más de un clúster, *`ClusterNamelist`* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *`ClusterName`* está en blanco. Al menos uno de *`ClusterName`* y *`ClusterNamelist`* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCHAR48)

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres a los que pertenece esta cola.

Tabla 571. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Si la cola sólo pertenece a un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *`ClusterName`* para especificar el nombre del clúster, en cuyo caso *`ClusterNamelist`* está en blanco. Al menos uno de *`ClusterName`* y *`ClusterNamelist`* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

CLWLQueuePriority (MQLONG)

Es la prioridad de cola de carga de trabajo de clúster, un valor comprendido entre 0 y 9 que representa la prioridad de la cola.

Tabla 572. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_PRIORITY con la llamada MQINQ.

CLWLQueueRank (MQLONG)

Este es el rango de cola de carga de trabajo de clúster, un valor en el rango de 0 a 9 que representa el rango de la cola.

Tabla 573. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_RANK con la llamada MQINQ.

CLWLUseQ (MQLONG)

Esto define el comportamiento de un MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remoto. Si la transferencia se origina en un canal de clúster, este atributo no es aplicable.

Tabla 574. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X				

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY
Utilice colas remotas y locales.

MQCLWL_USEQ_LOCAL
No utilice colas remotas.

MQCLWL_USEQ_AS_Q_MGR
Heredar definición de MQIA_CLWL_USEQ del gestor de colas.

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_USEQ con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)

Es la fecha en la que se creó la cola.

Tabla 575. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X				

El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para hacer que la longitud sea de 12 bytes (por ejemplo, `2013-09-23¬¬`, donde ¬¬ representa 2 caracteres en blanco).

- En IBM i, la fecha de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)

Es la hora en que se ha creado la cola.

Tabla 576. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

El formato de la hora es `HH.MM.SS` utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, `09.10.20`).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.
- En IBM i, la hora de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)

Es el número de mensajes que hay en la cola actualmente.

Tabla 577. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

Se incrementa durante una llamada MQPUT y durante la restitución de una llamada MQGET. Se reduce durante una llamada MQGET no examinada y durante la restitución de una llamada MQPUT. El efecto de esto es que el recuento incluye los mensajes que se han colocado en la cola dentro de una unidad de trabajo, pero que todavía no se han confirmado, aunque no sean elegibles para ser recuperados por la llamada MQGET. De forma similar, excluye los mensajes que se han recuperado dentro de una unidad de trabajo utilizando la llamada MQGET, pero que todavía no se han confirmado.

El recuento también incluye los mensajes que han pasado su hora de caducidad pero que todavía no se han descartado, aunque estos mensajes no son aptos para ser recuperados. Consulte Campo MQMD-Caducidad para obtener más información.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que *`CurrentQDepth`* supere *`MaxQDepth`*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes; *todos* los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CURRENT_Q_DEPTH con la llamada MQINQ.

Respuesta DefaultPut(MQLONG)

Especifica el tipo de respuesta que debe utilizarse para las operaciones de colocación en la cola cuando una aplicación especifica MQPMO_RESPONSE_AS_Q_DEF.

Tabla 578. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	

Es uno de los valores siguientes:

MQPRT_SYNC_RESPONSE
La operación de colocación se emite de forma síncrona, devolviendo una respuesta.

MQPRT_ASYNC_RESPONSE
La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

DefBind (MQLONG)

Este es el enlace predeterminado que se utiliza cuando se especifica MQOO_BIND_AS_Q_DEF en la llamada MQOPEN y la cola es una cola de clúster.

Tabla 579. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

El valor puede ser uno de los siguientes:

MQBND_BIND_ON_OPEN
Enlace arreglado por la llamada MQOPEN.

MQBND_BIND_NOT_FIXED
Enlace no arreglado.

MQBND_BIND_ON_GROUP
Permite a una aplicación solicitar que un grupo de mensajes se asigne a la misma instancia de destino. Puesto que este valor es nuevo en IBM WebSphere MQ 7.1, no se debe utilizar si alguna de las aplicaciones que abren esta cola se conecta a IBM WebSphere MQ 7.0.1 o a gestores de colas anteriores.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_BIND con la llamada MQINQ.

DefinitionType (MQLONG)

Indica cómo se ha definido la cola.

Tabla 580. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQDT_PREDEFINED
La cola es una cola permanente creada por el administrador del sistema; sólo el administrador del sistema puede suprimirla.

Las colas predefinidas se crean utilizando el mandato MQSC de `DEFINE` y solo se pueden suprimir utilizando el mandato MQSC de `DELETE` . Las colas predefinidas no se pueden crear a partir de colas modelo.

Los mandatos pueden ser emitidos por un operador o por un usuario autorizado que envía un mensaje de mandato a la cola de entrada de mandatos (consulte CommandInputQName para obtener más información).

MQQDT_PERMANENT_DYNAMIC
La cola es una cola permanente creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_PERMANENT_DYNAMIC para el atributo **`DefinitionType`** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte “MQCLOSE-Cerrar objeto” en la página 657 para obtener más detalles.

El valor del atributo **`QSGDisp`** para una cola dinámica permanente es MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC
La cola es una cola temporal creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_TEMPORARY_DYNAMIC para el atributo **DefinitionType** .

Este tipo de cola se suprime automáticamente mediante la llamada MQCLOSE cuando la aplicación que la ha creado la cierra.

El valor del atributo **QSGDisp** para una cola dinámica temporal es MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC
La cola es una cola permanente compartida creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_SHARED_DYNAMIC para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte "MQCLOSE-Cerrar objeto" en la página 657 para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica compartida es MQQSGD_SHARED.

Este atributo en una definición de cola modelo no indica cómo se ha definido la cola modelo, porque las colas modelo siempre están predefinidas. En su lugar, el valor de este atributo en la cola modelo se utiliza para determinar el *DefinitionType* de cada una de las colas dinámicas creadas a partir de la definición de cola modelo utilizando la llamada MQOPEN.

Para determinar el valor de este atributo, utilice el selector MQIA_DEFINITION_TYPE con la llamada MQINQ.

DefInputOpenOption (MQLONG)
Esta es la forma predeterminada en la que se abre la cola para la entrada.

Tabla 581. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Se aplica si se especifica la opción MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN cuando se abre la cola. El valor puede ser uno de los siguientes:

MQOO_INPUT_EXCLUSIVE
Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente por esta u otra aplicación para cualquier tipo de entrada (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_INPUT_OPEN_OPTION con la llamada MQINQ.

DefPersistence (MQLONG)
Es la persistencia predeterminada de los mensajes en la cola. Se aplica si se especifica MQPER_PERSISTENCE_AS_Q_DEF en el descriptor de mensaje cuando se coloca el mensaje.

Tabla 582. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la llamada MQPUT o MQPUT1 . Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

El valor puede ser uno de los siguientes:

MQPER_PERSISTENT
El mensaje sobrevive a las anomalías del sistema y se reinicia el gestor de colas. Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas que se correlacionan con un objeto CFSTRUCT en CFLEVEL (2) o inferior, o donde el objeto CFSTRUCT se define como RECOVER (NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

MQPER_NOT_PERSISTENT
Normalmente, el mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante un reinicio del gestor de colas.

En el caso de colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a los errores del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PERSISTENCE con la llamada MQINQ.

DefPriority (MQLONG)

Esta es la prioridad predeterminada para los mensajes de la cola. Esto se aplica si se especifica MQPRI_PRIORITY_AS_Q_DEF en el descriptor de mensaje cuando el mensaje se coloca en la cola.

Tabla 583. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada para el mensaje se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la operación de colocación. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas

- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

La forma en que se coloca un mensaje en una cola depende del valor del atributo **MsgDeliverySequence** de la cola:

- Si el atributo **MsgDeliverySequence** es MQMDS_PRIORITY, la posición lógica en la que se coloca un mensaje en la cola depende del valor del campo *Priority* en el descriptor de mensaje.
- Si el atributo **MsgDeliverySequence** es MQMDS_FIFO, los mensajes se colocan en la cola como si tuvieran una prioridad igual a la *DefPriority* de la cola resuelta, independientemente del valor del campo *Priority* en el descriptor de mensaje. Sin embargo, el campo *Priority* conserva el valor especificado por la aplicación que ha colocado el mensaje. Consulte Atributo de secuenciaMsgDelivery para obtener más información.

Las prioridades están en el rango de cero (menor) a *MaxPriority* (mayor); consulte el atributo MaxPriority.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PRIORITY con la llamada MQINQ.

DefReadpor omisión (MQLONG)

Especifica el comportamiento de lectura anticipada predeterminado para los mensajes no persistentes entregados al cliente.

Tabla 584. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

DefReadSe puede establecer en uno de los valores siguientes:

MQREADA_NO
: Los mensajes no persistentes no se envían al cliente antes de que las aplicaciones los soliciten. Puede perderse un mensaje no persistente como máximo, si el cliente finaliza de forma anómala.

MQREADA_SÍ
: Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Los mensajes no persistentes se pueden perder si el cliente finaliza de forma anómala o si el cliente no consume todos los mensajes que se envían.

MQREADA_DISABLED
: Lectura anticipada de mensajes no persistentes en no habilitados para esta cola. Los mensajes no se envían al cliente independientemente de si la aplicación cliente solicita la lectura anticipada.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_READ_AHEAD con la llamada MQINQ.

DefPResp (MQLONG)

El atributo de tipo de respuesta de colocación predeterminada (DEFPRESP) define el valor utilizado por las aplicaciones cuando el tipo PutResponsedentro de MQPMO se ha establecido en MQPMO_RESPONSE_AS_Q_DEF. Este atributo es válido para todos los tipos de cola.

Tabla 585. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

El valor puede ser uno de los siguientes:

SYNC
: La operación de colocación se emite de forma síncrona devolviendo una respuesta.

ASYNC
La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PUT_RESPONSE_TYPE con la llamada MQINQ.

DistLists (MQLONG)

Indica si los mensajes de lista de distribución se pueden colocar en la cola.

Tabla 586. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un agente de canal de mensajes (MCA) establece el atributo para informar al gestor de colas local si el gestor de colas del otro extremo del canal soporta listas de distribución. Este último gestor de colas (denominado gestor de colas de *asociación*) es el que recibe a continuación el mensaje, después de que un MCA emisor lo haya eliminado de la cola de transmisión local.

El MCA emisor establece el atributo siempre que establece una conexión con el MCA receptor en el gestor de colas asociado. De este modo, el MCA emisor puede hacer que el gestor de colas local coloque en la cola de transmisión sólo los mensajes que el gestor de colas asociado puede procesar correctamente.

Este atributo se utiliza principalmente con colas de transmisión, pero el proceso descrito se realiza independientemente del uso definido para la cola (consulte Atributo de uso).

El valor puede ser uno de los siguientes:

MQDL_SUPPORTED
Los mensajes de lista de distribución pueden almacenarse en la cola y transmitirse al gestor de colas asociado en ese formato. Esto reduce la cantidad de proceso necesaria para enviar el mensaje a varios destinos.

MQDL_NOT_SUPPORTED
Los mensajes de lista de distribución no se pueden almacenar en la cola porque el gestor de colas asociado no da soporte a las listas de distribución. Si una aplicación coloca un mensaje de lista de distribución y ese mensaje se va a colocar en esta cola, el gestor de colas divide el mensaje de lista de distribución y coloca los mensajes individuales en la cola. Esto aumenta la cantidad de proceso necesario para enviar el mensaje a varios destinos, pero garantiza que el gestor de colas asociado procese correctamente los mensajes.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Este atributo no está soportado en z/OS.

Restitución de HardenGet(MQLONG)

Para cada mensaje, se mantiene un recuento del número de veces que una llamada MQGET recupera el mensaje dentro de una unidad de trabajo, y dicha unidad de trabajo se restituye posteriormente.

Tabla 587. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este recuento está disponible en el campo `BackoutCount` del descriptor de mensaje después de que se haya completado la llamada MQGET.

El recuento de restituciones de mensajes sobrevive a los reinicios del gestor de colas. Sin embargo, para asegurarse de que el recuento es preciso, la información debe *guardarse* (grabarse en disco u otro dispositivo de almacenamiento permanente) cada vez que una llamada MQGET recupera un mensaje

dentro de una unidad de trabajo para esta cola. Si esto no se hace, el gestor de colas falla y la llamada MQGET se restituye, el recuento puede o no incrementarse.

Sin embargo, el endurecimiento de la información para cada llamada MQGET dentro de una unidad de trabajo impone un coste de proceso adicional, por lo tanto, establezca el atributo **HardenGetBackout** en MQQA_BACKOUT_HARAPARTADO sólo si es esencial que el recuento sea preciso.

En IBM i, UNIXy Windows, el recuento de restituciones de mensajes siempre está protegido, independientemente del valor de este atributo.

Son posibles los siguientes valores:

MQQA_BACKOUT_HARDENED
El refuerzo se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso.

MQQA_BACKOUT_NOT_HARTIZADO
El refuerzo no se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso. Por lo tanto, el recuento puede ser menor de lo que debería ser.

Para determinar el valor de este atributo, utilice el selector MQIA_HARDEN_GET_BACKOUT con la llamada MQINQ.

IndexType (MQLONG)

Especifica el tipo de índice que el gestor de colas mantiene para los mensajes de la cola.

Tabla 588. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El tipo de índice necesario depende de cómo la aplicación recupera los mensajes y de si la cola es una cola compartida o no compartida (consulte Atributo QSGDisp). Los valores siguientes son posibles para *IndexType*:

MQIT_NONE
El gestor de colas no mantiene ningún índice para esta cola. Utilice este valor para las colas que normalmente se procesan secuencialmente, es decir, sin utilizar ningún criterio de selección en la llamada MQGET.

ID_MSG_MQIT
El gestor de colas mantiene un índice que utiliza los identificadores de mensaje de los mensajes de la cola. Utilice este valor en las colas en las que la aplicación normalmente recupera mensajes utilizando el identificador de mensaje como criterio de selección en la llamada MQGET.

ID_CORREL_MQIT
El gestor de colas mantiene un índice que utiliza los identificadores de correlación de los mensajes de la cola. Utilice este valor para las colas en las que la aplicación normalmente recupera mensajes utilizando el identificador de correlación como criterio de selección en la llamada MQGET.

MQIT_MSG_TOKEN

Importante: Este tipo de índice sólo se debe utilizar para las colas utilizadas con el producto IBM MQ Workflow for z/OS .

El gestor de colas mantiene un índice que utiliza las señales de mensaje de los mensajes de la cola para su uso con las funciones del gestor de carga de trabajo (WLM) de z/OS.

Debe especificar esta opción para colas gestionadas por WLM; no la especifique para ningún otro tipo de cola. Además, no utilice este valor para una cola en la que una aplicación no está utilizando las funciones del gestor de carga de trabajo de z/OS , sino que está recuperando mensajes utilizando la señal de mensaje como criterio de selección en la llamada MQGET.

ID_grupo_MQIT

El gestor de colas mantiene un índice que utiliza los identificadores de grupo de los mensajes de la cola. Este valor debe utilizarse para las colas en las que la aplicación recupera mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.

Una cola con este tipo de índice no puede ser una cola de transmisión. Debe definirse una cola compartida con este tipo de índice para correlacionar con un objeto CFSTRUCT en CFLEVEL (3) o superior.

Nota:

1. El orden físico de los mensajes en una cola con el tipo de índice MQIT_GROUP_ID no está definido, ya que la cola está optimizada para la recuperación eficaz de mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET. Esto significa que el orden físico de los mensajes no suele ser el orden en el que los mensajes han llegado a la cola.
2. Si una cola MQIT_GROUP_ID tiene un *MsgDeliverySequence* de MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensajes 0 y 1 para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si lo tiene, el mensaje se procesa como si tuviera una prioridad de dos. El campo *Priority* de la estructura MQMD no se modifica.

Para obtener más información sobre los grupos de mensajes, consulte la descripción de las opciones de grupo y segmento en el campo MQGMO-Opciones.

El tipo de índice que se debe utilizar en varios casos se muestra en Tabla 589 en la página 865 y Tabla 590 en la página 866.

Tabla 589. Valores sugeridos o necesarios del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER

Criterios de selección en la llamada MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Ninguna	Cualquiera	Cualquiera
Selección utilizando un identificador:		
Identificador del mensaje	MQIT_MSG_ID sugerido	MQIT_NONE o MQIT_MSG_ID necesarios; MQIT_MSG_ID sugerido
Identificador de correlación	MQIT_CORREL_ID sugerido	MQIT_CORREL_ID necesario
Identificador de grupo	MQIT_GROUP_ID sugerido	MQIT_GROUP_ID necesario
Selección utilizando dos identificadores:		
Identificador de mensaje más identificador de correlación	MQIT_MSG_ID o MQIT_CORREL_ID sugeridos	MQIT_NONE o MQIT_MSG_ID o MQIT_CORREL_ID necesarios (Para mayor eficacia, se sugiere que el tipo de índice se elija para que coincida con el campo MQMD que tendrá las claves más diferenciadas)
Identificador de mensaje más identificador de grupo	MQIT_MSG_ID o MQIT_GROUP_ID sugeridos	No soportado
Identificador de correlación más identificador de grupo	MQIT_CORREL_ID o MQIT_GROUP_ID sugeridos	No soportado
Selección utilizando tres identificadores:		

Tabla 589. Valores sugeridos o necesarios del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER (continuación)		
Criterios de selección en la llamada MQGET	**Tipo de índice para cola no compartida**	**Tipo de índice para cola compartida**
Identificador de mensaje más identificador de correlación más identificador de grupo	MQIT_MSG_ID o MQIT_CORREL_ID o MQIT_GROUP_ID sugeridos	No soportado
Selección utilizando criterios relacionados con grupos:		
Identificador de grupo más número de secuencia de mensaje	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Número de secuencia de mensaje (debe ser 1)	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Selección utilizando señal de mensaje:		
Señal de mensaje para uso de la aplicación	No utilizar MQIT_MSG_TOKEN	
Señal de mensaje para uso de WLM	MQIT_MSG_TOKEN necesario	No soportado

Tabla 590. Valores sugeridos o necesarios del tipo de índice de cola cuando se especifica MQGMO_LOGICAL_ORDER		
Criterios de selección en la llamada MQGET	**Tipo de índice para cola no compartida**	**Tipo de índice para cola compartida**
Ninguna	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Selección utilizando un identificador:		
Identificador del mensaje	MQIT_GROUP_ID necesario	No soportado
Identificador de correlación	MQIT_GROUP_ID necesario	No soportado
Identificador de grupo	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Selección utilizando dos identificadores:		
Identificador de mensaje más identificador de correlación	MQIT_GROUP_ID necesario	No soportado
Identificador de mensaje más identificador de grupo	MQIT_GROUP_ID necesario	No soportado
Identificador de correlación más identificador de grupo	MQIT_GROUP_ID necesario	No soportado
Selección utilizando tres identificadores:		
Identificador de mensaje más identificador de correlación más identificador de grupo	MQIT_GROUP_ID necesario	No soportado

Para determinar el valor de este atributo, utilice el selector MQIA_INDEX_TYPE con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

InhibitGet (MQLONG)

Esto controla si se permiten operaciones get para esta cola.

Tabla 591. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X		

Si la cola es una cola alias, las operaciones get deben estar permitidas tanto para el alias como para la cola base en el momento de la operación get, para que la llamada MQGET sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_GET_INHIBITED
Las operaciones de obtención están inhibidas.

Las llamadas MQGET fallan con el código de razón MQRC_GET_inhibiITED. Esto incluye las llamadas MQGET que especifican MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Nota: Si una llamada MQGET que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **`InhibitGet`** posteriormente a MQQA_GET_inhibiITED no impide que se confirme la unidad de trabajo.

MQQA_GET_ALLOWED
Las operaciones de obtención están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_GET con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InhibitPut (MQLONG)

Esto controla si se permiten las operaciones de colocación para esta cola.

Tabla 592. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, se deben permitir operaciones de colocación para *cada* definición en la vía de acceso (incluidas las definiciones de alias de gestor de colas) en el momento de la operación de colocación, para que la llamada MQPUT o MQPUT1 sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_PUT_INHIBITED
Las operaciones de colocación están inhibidas.

Las llamadas MQPUT y MQPUT1 fallan con el código de razón MQRC_PUT_inhibiITED.

Nota: Si una llamada MQPUT que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **`InhibitPut`** posteriormente a MQQA_PUT_inhibiITED no impide que se confirme la unidad de trabajo.

MQQA_PUT_ALLOWED
Las operaciones de colocación están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_PUT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InitiationQName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local; la cola debe ser de tipo MQQT_LOCAL.

Tabla 593. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

El gestor de colas envía un mensaje desencadenante a la cola de inicio cuando es necesario iniciar la aplicación como resultado de un mensaje que llega a la cola a la que pertenece este atributo. La cola de inicio debe ser supervisada por una aplicación de supervisor desencadenante que inicie la aplicación adecuada después de recibir el mensaje desencadenante.

Para determinar el valor de este atributo, utilice el selector MQCA_INITIATION_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

MaxMsgLongitud (MQLONG)

Este es un límite superior para la longitud del mensaje *físico* más largo que se puede colocar en la cola.

Tabla 594. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Sin embargo, debido a que el atributo de cola **MaxMsgLength** se puede establecer independientemente del atributo de gestor de colas **MaxMsgLength** , el límite superior real para la longitud del mensaje físico más largo que se puede colocar en la cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength** , pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

Un intento de colocar en la cola un mensaje demasiado largo falla con uno de los siguientes códigos de razón:

- MQRC_MSG_TOO_BIG_FOR_Q si el mensaje es demasiado grande para la cola
- MQRC_MSG_TOO_BIG_FOR_Q_MGR si el mensaje es demasiado grande para el gestor de colas, pero no demasiado grande para la cola

El límite inferior para el atributo **MaxMsgLength** es cero; el límite superior es de 100 MB (104 857 600 bytes).

Para obtener más información, consulte MQPUT-Parámetro BufferLength.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxQDepth (MQLONG)

Es el límite superior definido para el número de mensajes físicos que pueden existir en la cola en cualquier momento.

Tabla 595. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Un intento de colocar un mensaje en una cola que ya contiene mensajes **MaxQDepth** falla con el código de razón MQRC_Q_FULL.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que el número real de mensajes físicos en la cola supere **MaxQDepth**. Sin embargo, esto no afecta a la capacidad de

recuperación del mensaje porque todos los mensajes de la cola se pueden recuperar utilizando la llamada MQGET.

El valor de este atributo es cero o mayor. El límite superior lo determina el entorno:

- En las plataformas siguientes, el valor no puede exceder de 999 999 999:
  - AIX
  - Linux
  - Solaris
  - Windows
  - z/OS
- IBM i En IBM i, el valor no puede superar los 640 000.

Nota: El espacio de almacenamiento disponible para la cola puede agotarse incluso si hay menos de **MaxQDepth** mensajes en la cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_Q_DEPTH con la llamada MQINQ.

Secuencia MsgDelivery(MQLONG)

Tabla 596. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Esto determina el orden en el que la llamada MQGET devuelve mensajes a la aplicación:

MQMDS_FIFO
Los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Una llamada MQGET devuelve el *primer* mensaje que cumple los criterios de selección especificados en la llamada, independientemente de la prioridad del mensaje.

MQMDS_PRIORITY
Los mensajes se devuelven en orden de prioridad.

Una llamada MQGET devuelve el mensaje de *prioridad más alta* que cumple los criterios de selección especificados en la llamada. Dentro de cada nivel de prioridad, los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

- En z/OS, si la cola tiene un *IndexType* de MQIT_GROUP_ID, el atributo **MsgDeliverySequence** especifica el orden en el que se devuelven los grupos de mensajes a la aplicación. La secuencia particular en la que se devuelven los grupos viene determinada por la posición o prioridad del primer mensaje de cada grupo. El orden físico de los mensajes en la cola no está definido, ya que la cola está optimizada para una recuperación eficaz de mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.
- En z/OS, si *IndexType* es MQIT_GROUP_ID y *MsgDeliverySequence* es MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensaje cero y una para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si lo tiene, el mensaje se procesa como si tuviera una prioridad de dos. El campo *Priority* de la estructura MQMD no se modifica.

Si los atributos relevantes se cambian mientras hay mensajes en la cola, la secuencia de entrega es la siguiente:

- El orden en el que la llamada MQGET devuelve los mensajes viene determinado por los valores de los atributos **MsgDeliverySequence** y **DefPriority** en vigor para la cola en el momento en que el mensaje llega a la cola:

– Si *MsgDeliverySequence* es MQMDS_FIFO cuando llega el mensaje, el mensaje se coloca en la cola como si su prioridad fuera *DefPriority*. Esto no afecta al valor del campo *Priority* en el descriptor de mensaje del mensaje; dicho campo conserva el valor que tenía cuando se colocó el mensaje por primera vez.
– Si *MsgDeliverySequence* es MQMDS_PRIORITY cuando llega el mensaje, el mensaje se coloca en la cola en el lugar adecuado a la prioridad proporcionada por el campo *Priority* en el descriptor de mensaje.

Si el valor del atributo **MsgDeliverySequence** cambia mientras hay mensajes en la cola, el orden de los mensajes en la cola no cambia.

Si el valor del atributo **DefPriority** se cambia mientras hay mensajes en la cola, los mensajes no se entregan necesariamente en orden FIFO, aunque el atributo **MsgDeliverySequence** se establezca en MQMDS_FIFO; los que se colocaron en la cola con la prioridad más alta se entregan primero.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_DELIVERY_SEQUENCE con la llamada MQINQ.

NonPersistentMessageClass (MQLONG)

El objetivo de fiabilidad para los mensajes no persistentes.

Tabla 597. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Especifica las circunstancias en las que se descartan los mensajes no persistentes colocados en esta cola:

MQNPM_CLASS_NORMAL
Los mensajes no persistentes están limitados al tiempo de vida de la sesión del gestor de colas; los mensajes se descartan en el caso de un reinicio del gestor de colas. Sólo es válido para colas no compartidas y es el valor predeterminado.

MQNPM_CLASS_HIGH
El gestor de colas intenta retener los mensajes no persistentes durante el tiempo de vida de la cola. Es posible que los mensajes no persistentes se pierdan si se produce una anomalía. Este valor se aplica a las colas compartidas.

Para determinar el valor de este atributo, utilice el selector MQIA_NPM_CLASS con la llamada MQINQ.

Recuento de OpenInput(MQLONG)

Es el número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola mediante la llamada MQGET.

Tabla 598. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* . Si la cola es una cola compartida, el recuento no incluye las aperturas para la entrada que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para entrada. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la entrada (por ejemplo, una cola abierta sólo para examinar).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_INPUT_COUNT con la llamada MQINQ.

Recuento de OpenOutput(MQLONG)

Es el número de descriptores de contexto que son válidos actualmente para añadir mensajes a la cola mediante la llamada MQPUT.

Tabla 599. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* ; no incluye las aperturas para salida que se han realizado para esta cola en los gestores de colas remotos. Si la cola es una cola compartida, el recuento no incluye las aperturas para la salida que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para salida. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la salida (por ejemplo, una cola abierta sólo para consulta).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_OUTPUT_COUNT con la llamada MQINQ.

ProcessName (MQCHAR48)

Es el nombre de un objeto de proceso definido en el gestor de colas local. El objeto de proceso identifica un programa que puede dar servicio a la cola.

Tabla 600. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Especifica cómo se manejan las propiedades de mensaje para los mensajes que se recuperan de las colas utilizando la llamada MQGET con la opción MQGMO_PROPERTIES_AS_Q_DEF.

Tabla 601. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

El valor puede ser uno de los siguientes:

MQPROP_ALL
Todas las propiedades del mensaje se incluyen con el mensaje cuando se entrega a la aplicación. Las propiedades, excepto las que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

COMPATIBILIDAD de MQPROP_COMPATIBILITY
Si el mensaje contiene una propiedad con el prefijo mcd., jms., usr. o mqext., todas las propiedades de mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación. Este es el valor predeterminado; permite que las

aplicaciones que esperan que las propiedades relacionadas con JMS estén en una cabecera MQRFH2 en los datos del mensaje sigan funcionando sin modificar. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

MQPROP_FORCE_MQRFH2
Las propiedades siempre se devuelven en los datos de mensaje en una cabecera MQRFH2 independientemente de si la aplicación especifica un manejador de mensajes. Se ignora un manejador de mensajes válido proporcionado en el campo MsgHandle de la estructura MQGMO en la llamada MQGET. Las propiedades del mensaje no son accesibles a través del manejador de mensajes.

MQPROP_NONE
Todas las propiedades del mensaje, excepto las del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se entregue a la aplicación. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

Este parámetro es aplicable a las colas Local, Alias y Modelo. Para determinar su valor, utilice el selector MQIA_PROPERTY_CONTROL con la llamada MQINQ.

Suceso QDepthHigh(MQLONG)

Esto controla si se generan sucesos de profundidad de cola alta.

Tabla 602. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso de profundidad de cola alta indica que una aplicación ha colocado un mensaje en una cola y esto ha hecho que el número de mensajes de la cola sea mayor o igual que el umbral de profundidad de cola alta (consulte el atributo **QDepthHighLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Límite QDepthHigh(MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola alta.

Tabla 603. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este suceso indica que una aplicación ha colocado un mensaje en una cola y que esto ha hecho que el número de mensajes en la cola sea mayor o igual que el umbral alto de profundidad de cola. Consulte Atributo de sucesoQDepthHigh.

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo**MaxQDepth**), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es 80.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Suceso QDepthLow(MQLONG)

Esto controla si se generan sucesos de profundidad de cola baja.

Tabla 604. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso Profundidad de cola baja indica que una aplicación ha recuperado un mensaje de una cola y que esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral de profundidad de cola baja (consulte atributo LímiteQDepthLow).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
: Informes de sucesos inhabilitados.

MQEVR_ENABLED
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Límite QDepthLow(MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola baja.

Tabla 605. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este suceso indica que una aplicación ha recuperado un mensaje de una cola y que esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral bajo de profundidad de cola. Consulte Atributo de sucesoQDepthLow.

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo**MaxQDepth**), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es de 20.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Suceso QDepthMax(MQLONG)

Esto controla si se generan sucesos de cola llena. Un suceso Cola llena indica que una colocación en una cola se ha rechazado porque la cola está llena, es decir, la profundidad de cola ya ha alcanzado su valor máximo.

Tabla 606. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_MAX_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos.

Tabla 607. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_DESC_LENGTH.

QName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local.

Tabla 608. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	X

Todas las colas definidas en un gestor de colas comparten el mismo espacio de nombres de cola. Por lo tanto, una cola MQQT_LOCAL y una cola MQQT_ALIAS no pueden tener el mismo nombre.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Este es el intervalo de servicio utilizado para la comparación para generar sucesos de intervalo de servicio alto y de intervalo de servicio correcto.

Tabla 609. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Consulte Atributo de sucesoQServiceInterval.

El valor está en unidades de milisegundos, y es mayor o igual que cero, y menor o igual que 999 999 999 999.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QServiceIntervalEvent (MQLONG)

Esto controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto.

Tabla 610. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

- Se genera un suceso de intervalo de servicio alto cuando una comprobación indica que no se ha recuperado ningún mensaje de la cola durante al menos el tiempo indicado por el atributo **`QServiceInterval`** .
- Se genera un suceso de intervalo de servicio correcto cuando una comprobación indica que los mensajes se han recuperado de la cola dentro del tiempo indicado por el atributo **`QServiceInterval`** .

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQQSIE_HIGH
Sucesos de intervalo de servicio de cola alto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **habilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **inhabilitados**.

MQQSIE_OK
Sucesos de intervalo de servicio de cola correcto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **habilitados**.

MQQSIE_NONE
No hay sucesos de intervalo de servicio de cola habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto también están **inhabilitados**.

Para las colas compartidas, el valor de este atributo se ignora; se asume el valor MQQSIE_NONE.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

QSGDisp (MQLONG)

Especifica la disposición de la cola.

Tabla 611. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene disposición de gestor de colas. Esto significa que la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero utilizando mandatos MQSC, puede modificar cada copia para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

MQQSGD_SHARED
El objeto tiene una disposición compartida. Esto significa que existe en el repositorio compartido una única instancia del objeto que es conocida por todos los gestores de colas del grupo de compartición de colas. Cuando un gestor de colas del grupo accede al objeto, accede a la única instancia compartida del objeto.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Tabla 612. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	

Esto controla la recopilación de datos de contabilidad para la cola. Para que los datos de contabilidad se recopilen para esta cola, los datos de contabilidad para esta conexión también deben estar habilitados, utilizando el atributo ACCTQ de QMGR o el campo Opciones de la estructura MQCNO en la llamada MQCONNX.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR
Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo ACCTQ de QMGR. Este es el valor predeterminado.

MQMON_OFF
No recopile datos de contabilidad para esta cola.

MQMON_ON
Recopilar datos de contabilidad para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Controla la recopilación de los datos de supervisión para las colas.

Tabla 613. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
Recopile datos de supervisión según el valor del atributo de gestor de colas **QueueMonitoring** . Este es el valor predeterminado.

MQMON_OFF
La recopilación de datos de supervisión en línea está desactivada para esta cola.

MQMON_LOW
Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una tasa baja de recopilación de datos para esta cola.

MQMON_MEDIO
Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, la recopilación de datos de supervisión en línea está activada, con una tasa moderada de recopilación de datos para esta cola.

MQMON_HIGH
Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una alta tasa de recopilación de datos para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQCHAR12)

Tabla 614. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	

Esto controla la recopilación de datos estadísticos para la cola.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR
Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo STATQ de QMGR. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para esta cola.

MQMON_ON
Habilite la recopilación de datos de estadísticas para esta cola.

Tipo de cola (MQLONG)

Tabla 615. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	X

Este es el tipo de cola; tiene uno de los valores siguientes:

MQQT_ALIAS
Definición de cola alias.

MQQT_CLUSTER
Cola de clúster.

MQQT_LOCAL
Cola local.

MQQT_REMOTE
Definición local de una cola remota.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_TYPE con la llamada MQINQ.

RemoteQMgrNombre (MQCHAR48)

Tabla 616. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
			X	

Es el nombre del gestor de colas remoto en el que se define la cola **RemoteQName** . Si la cola **RemoteQName** tiene un valor **QSGDisp** de MQQSGD_COPY o MQQSGD_SHARED, **RemoteQMgrName** puede ser el nombre del grupo de compartición de colas propietario de **RemoteQName**.

Si una aplicación abre la definición local de una cola remota, **RemoteQMgrName** no debe estar en blanco y no debe ser el nombre del gestor de colas local. Si **XmitQName** está en blanco, se utiliza la cola local con el mismo nombre que **RemoteQMgrName** como cola de transmisión. Si no hay ninguna cola con el nombre **RemoteQMgrName**, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName** .

Si esta definición se utiliza para un alias de gestor de colas, **RemoteQMgrName** es el nombre del gestor de colas que se está alias. Puede ser el nombre del gestor de colas local. De lo contrario, si **XmitQName** está en blanco cuando se produce la apertura, debe haber una cola local con un nombre que sea el mismo que **RemoteQMgrName**; esta cola se utiliza como cola de transmisión.

Si esta definición se utiliza para un alias de respuesta, este nombre es el nombre del gestor de colas que debe ser **ReplyToQMgr**.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Tabla 617. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
			X	

Es el nombre de la cola tal como se conoce en el gestor de colas remoto *RemoteQMgrName*.

Si una aplicación abre la definición local de una cola remota, cuando se produce la apertura, *RemoteQName* no debe estar en blanco.

Si esta definición se utiliza para una definición de alias de gestor de colas, cuando se produzca la apertura, *RemoteQName* debe estar en blanco.

Si la definición se utiliza para un alias de respuesta, este nombre es el nombre de la cola que va a ser *ReplyToQ*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Es el periodo de tiempo durante el cual se retiene la cola. Una vez transcurrido este tiempo, la cola es apta para su supresión.

Tabla 618. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El tiempo se mide en horas, contando desde la fecha y hora en que se creó la cola. La fecha y hora de creación de la cola se registran en los atributos **CreationDate** y **CreationTime** .

Esta información se proporciona para permitir que una aplicación de mantenimiento o el operador identifique y suprima colas que ya no son necesarias.

Nota: El gestor de colas nunca realiza ninguna acción para suprimir colas basándose en este atributo, o para impedir la supresión de colas con un intervalo de retención que no ha caducado; es responsabilidad del usuario realizar cualquier acción necesaria.

Utilice un intervalo de retención realista para evitar la acumulación de colas dinámicas permanentes (consulte el atributo DefinitionType). Sin embargo, este atributo también se puede utilizar con colas predefinidas.

Para determinar el valor de este atributo, utilice el selector MQIA_RETENTION_INTERVAL con la llamada MQINQ.

Ámbito (MQLONG)

Esto controla si también existe una entrada para esta cola en un directorio de célula.

Tabla 619. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Un servicio de nombres instalable proporciona un directorio de célula. El valor puede ser uno de los siguientes:

MQSCO_Q_MGR
La definición de cola tiene ámbito de gestor de colas: la definición de la cola no se extiende más allá del gestor de colas que la posee. Para abrir la cola para salida de algún otro gestor de colas, debe especificarse el nombre del gestor de colas propietario o el otro gestor de colas debe tener una definición local de la cola.

MQSCO_CELL
La definición de cola tiene ámbito de célula: la definición de cola también se coloca en un directorio de célula disponible para todos los gestores de colas de la célula. La cola se puede abrir para salida de cualquiera de los gestores de colas de la célula especificando el nombre de la cola; no es necesario especificar el nombre del gestor de colas propietario de la cola. Sin embargo, la definición de cola no está disponible para ningún gestor de colas de la célula que también tenga una definición local de una cola con ese nombre, ya que la definición local tiene prioridad.

Un servicio de nombres instalable proporciona un directorio de célula.

El modelo y las colas dinámicas no pueden tener ámbito de célula.

Este valor sólo es válido si se ha configurado un servicio de nombres que da soporte a un directorio de célula.

Para determinar el valor de este atributo, utilice el selector MQIA_SCOPE con la llamada MQINQ.

El soporte para este atributo está sujeto a las restricciones siguientes:

- En IBM i, el atributo está soportado, pero sólo MQSCO_Q_MGR es válido.
- En z/OS, el atributo no está soportado.

Compartibilidad (MQLONG)

Esto indica si la cola se puede abrir para entrada varias veces simultáneamente.

Tabla 620. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQA_SHAREABLE
La cola es compartible.

Se permiten varias aperturas con la opción MQOO_INPUT_SHARED.

MQQA_NOT_SHAREABLE
La cola no se puede compartir.

Una llamada MQOPEN con la opción MQOO_INPUT_SHARED se trata como MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_SHAREABILITY con la llamada MQINQ.

StorageClass (MQCHAR8)

Es un nombre definido por el usuario que define el almacenamiento físico utilizado para contener la cola. En la práctica, un mensaje se graba en disco sólo si necesita paginarse fuera de su almacenamiento intermedio de memoria.

Tabla 621. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_STORAGE_CLASS con la llamada MQINQ. La longitud de este atributo la proporciona MQ_STORAGE_CLASS_LENGTH.

z/OS Este atributo sólo está soportado en z/OS.

TriggerControl (MQLONG)

Esto controla si los mensajes desencadenantes se graban en una cola de inicio para iniciar una aplicación para dar servicio a la cola.

Tabla 622. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es uno de los siguientes:

MQTC_OFF
No se deben escribir mensajes desencadenantes para esta cola. El valor de *TriggerType* es irrelevante en este caso.

MQTC_ON
Los mensajes desencadenantes se escribirán para esta cola cuando se produzcan los sucesos desencadenantes adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_CONTROL con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerData (MQCHAR64)

Son datos de formato libre que el gestor de colas inserta en el mensaje desencadenante cuando un mensaje que llega a esta cola hace que se grabe un mensaje desencadenante en la cola de inicio.

Tabla 623. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El contenido de estos datos no es significativo para el gestor de colas. Es significativo para la aplicación de supervisor desencadenante que procesa la cola de inicio o para la aplicación que inicia el supervisor desencadenante.

La serie de caracteres no debe contener ningún valor nulo. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_TRIGGER_DATA con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET. La longitud de este atributo la proporciona MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Tabla 624. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el número de mensajes de prioridad *TriggerMsgPriority* o superior que deben estar en la cola antes de que se escriba un mensaje desencadenante. Esto se aplica cuando *TriggerType* se establece en MQTT_DEPTH. El valor de *TriggerDepth* es uno o mayor. De lo contrario, este atributo no se utiliza.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_DEPTH con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerMsgPriority (MQLONG)

Esta es la prioridad de mensaje por debajo de la cual los mensajes no contribuyen a la generación de mensajes desencadenantes (es decir, el gestor de colas ignora estos mensajes al decidir si se genera un mensaje desencadenante).

Tabla 625. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

TriggerMsgPriority puede estar en el rango de cero (más bajo) a *MaxPriority* (más alto; consulte el atributo MaxPriority); un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_MSG_PRIORITY con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerType (MQLONG)

Esto controla las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola.

Tabla 626. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Tiene uno de los siguientes valores:

MQTT_NONE
No se graban mensajes desencadenantes como resultado de los mensajes de esta cola. Esto tiene el mismo efecto que establecer *TriggerControl* en MQTC_OFF.

MQTT_FIRST
Se graba un mensaje desencadenante siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola cambia de 0 a 1.

MQTT_EVERY
Se graba un mensaje desencadenante siempre que llega a la cola un mensaje de prioridad *TriggerMsgPriority* o superior.

MQTT_DEPTH
Un mensaje desencadenante se graba siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola es igual o superior a *TriggerDepth*. Después de que se haya grabado el mensaje desencadenante, *TriggerControl* se establece en MQTC_OFF para evitar que se desencadene más hasta que se vuelva a activar explícitamente.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_TYPE con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Uso (MQLONG)

Indica para qué se utiliza la cola.

Tabla 627. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQUS_NORMAL
Se trata de una cola que las aplicaciones utilizan al transferir y obtener mensajes; la cola no es una cola de transmisión.

MQUS_TRANSMISSION
Es una cola utilizada para contener mensajes destinados a gestores de colas remotos. Cuando una aplicación envía un mensaje a una cola remota, el gestor de colas local almacena el mensaje temporalmente en la cola de transmisión adecuada en un formato especial. A continuación, un agente de canal de mensajes lee el mensaje de la cola de transmisión y lo transporta al gestor de colas remoto. Para obtener más información sobre cómo configurar la administración remota, consulte Configuración de gestores de colas para la administración remota.

Sólo las aplicaciones con privilegios pueden abrir una cola de transmisión para que MQOO_OUTPUT coloque mensajes directamente en ella. Normalmente, sólo las aplicaciones de programa de utilidad lo hacen. Asegúrese de que el formato de datos del mensaje sea correcto (consulte “MQXQH-Cabecera de cola de transmisión” en la página 625) o se pueden producir errores durante el proceso

de transmisión. El contexto no se pasa ni se establece a menos que se especifique una de las opciones de contexto MQPMO_ * _CONTEXT.

Para determinar el valor de este atributo, utilice el selector MQIA_USAGE con la llamada MQINQ.

XmitQName (MQCHAR48)

Es el nombre de la cola de transmisión. Si este atributo no está en blanco cuando se produce una apertura, ya sea para una cola remota o para una definición de alias de gestor de colas, especifica el nombre de la cola de transmisión local que se utilizará para reenviar el mensaje.

Tabla 628. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
			X	

Si **XmitQName** está en blanco, la cola local con un nombre que es el mismo que **RemoteQMgrName** se utiliza como cola de transmisión. Si no hay ninguna cola con el nombre **RemoteQMgrName**, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName** .

Este atributo se ignora si la definición se está utilizando como alias de gestor de colas y **RemoteQMgrName** es el nombre del gestor de colas local. Este atributo también se ignora si la definición se utiliza como definición de alias de cola de respuestas.

Para determinar el valor de este atributo, utilice el selector MQCA_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

Atributos de las listas de nombres

En la tabla siguiente se resumen los atributos específicos de las listas de nombres. Los atributos se describen en orden alfabético.

Las listas de nombres están soportadas en todos los sistemas IBM MQ , más IBM MQ MQI clients conectados a estos sistemas.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más información.

Tabla 629. Atributos de las listas de nombres

Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
NameCount	Número de nombres en la lista de nombres
NamelistDesc	Descripción de lista de nombres
NamelistName	Nombre de lista de nombres
Nombres	Una lista de nombres de *NameCount*
NamelistType	Tipo de lista de nombres
QSGDisp	Disposición de grupo de uso compartido de colas

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

NameCount (MQLONG)

Es el número de nombres de la lista de nombres. Es mayor o igual que cero. Se define el valor siguiente:

MQNC_MAX_NAMELIST_NAME_COUNT
: Número máximo de nombres en una lista de nombres.

Para determinar el valor de este atributo, utilice el selector MQIA_NAME_COUNT con la llamada MQINQ.

NamelistDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos; su valor se establece mediante el proceso de definición. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **`CodedCharSetId`**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Es el nombre de una lista de nombres definida en el gestor de colas local. Para obtener más información sobre los nombres de lista de nombres, consulte la sección Otros nombres de objeto .

Cada lista de nombres tiene un nombre que es diferente de los nombres de otras listas de nombres que pertenecen al gestor de colas, pero puede duplicar los nombres de otros objetos del gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Especifica la naturaleza de los nombres de la lista de nombres e indica cómo se utiliza la lista de nombres. Es uno de los valores siguientes:

MQNT_NONE
: Lista de nombres sin tipo asignado.

MQNT_Q
: Lista de nombres que contiene los nombres de las colas.

MQNT_CLUSTER
: Lista de nombres que contiene los nombres de los clústeres.

MQNT_AUTH_INFO
: Lista de nombres que contiene los nombres de los objetos de información de autenticación.

Para determinar el valor de este atributo, utilice el selector MQIA_NAMELIST_TYPE con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

Nombres (MQCHAR48xNameCount)

Esta es una lista de nombres de *NameCount* , donde cada nombre es el nombre de un objeto definido en el gestor de colas local. Para obtener más información sobre los nombres de objeto, consulte Reglas para la denominación de objetos de IBM MQ.

Para determinar el valor de este atributo, utilice el selector MQCA_NAMES con la llamada MQINQ.

La longitud de cada nombre de la lista la proporciona MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la lista de nombres. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia, utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

Atributos de las definiciones de proceso

En la tabla siguiente se resumen los atributos que son específicos de las definiciones de proceso. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos de esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más información.

Tabla 630. Atributos de las definiciones de proceso

Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
ApplId	Identificador de aplicación
ApplType	Tipo de aplicación
EnvData	Datos de entorno
ProcessDesc	Descripción del proceso
ProcessName	Nombre de proceso
QSGDisp	Disposición de grupo de uso compartido de colas
UserData	Datos de usuario

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Es una serie de caracteres que identifica la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *ApplId* sea el nombre de un programa ejecutable. Las notas siguientes se aplican a los entornos indicados:

- En z/OS, *ApplId* debe ser:
  - Un identificador de transacción CICS , para las aplicaciones iniciadas utilizando la transacción `CKTI` del supervisor desencadenante de CICS
  - Un identificador de transacción IMS , para las aplicaciones iniciadas utilizando el supervisor desencadenante de IMS CSQQTRMN
- En sistemas Windows , el nombre de programa puede tener como prefijo una unidad y una vía de acceso de directorio.
- En UNIX, el nombre de programa puede tener como prefijo una vía de acceso de directorio.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_APPL_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Identifica la naturaleza del programa que se va a iniciar en respuesta a la recepción de un mensaje desencadenante. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

ApplType puede tener cualquier valor, pero se recomiendan los siguientes valores para los tipos estándar; restrinja los tipos de aplicación definidos por el usuario a los valores del rango MQAT_USER_FIRST a MQAT_USER_LAST:

MQAT_AIX
Aplicación AIX (mismo valor que MQAT_UNIX).

MQAT_LOTE
aplicación por lotes

INTERMEDIARIO
Aplicación de intermediario

MQAT_CICS
Transacción CICS .

MQAT_CICS_PUENTE
CICS bridge .

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
Aplicación IBM MQ MQI client en PC DOS.

MQAT_IMS
IMS .

MQAT_IMS_BRIDGE
Aplicación puente IMS .

MQAT_JAVA
Java .

MQAT_MVS
Aplicación MVS o TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Aplicación de agente.

MQAT_OS390
Aplicación OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
IBM i .

MQAT_RRS_BATCH
Aplicación por lotes RRS.

MQAT_UNIX
UNIX .

MQAT_DESCONOCIDO
Aplicación de tipo desconocido.

USUARIO_MQ
Aplicación de usuario.

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
z/OS .

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

Para determinar el valor de este atributo, utilice el selector MQIA_APPL_TYPE con la llamada MQINQ.

EnvData (MQCHAR128)

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *EnvData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ añade *EnvData* a la lista de parámetros pasada a la aplicación iniciada. La lista de parámetros consta de la estructura MQTMC2 , seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados. Las notas siguientes se aplican a los entornos indicados:

- En z/OS:
  - *EnvData* no es utilizado por las aplicaciones de supervisor desencadenante proporcionadas por IBM MQ.
  - Si ApplType es MQAT_WLM, puede proporcionar valores predeterminados en EnvData para los campos ServiceName y ServiceStep en la cabecera de información de trabajo (MQWIH).
- En UNIX, *EnvData* se puede establecer en el carácter & para ejecutar la aplicación iniciada en segundo plano.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_ENV_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Es el nombre de una definición de proceso definida en el gestor de colas local.

Cada definición de proceso tiene un nombre que es diferente de los nombres de otras definiciones de proceso que pertenecen al gestor de colas. Pero el nombre de la definición de proceso puede ser el mismo que los nombres de otros objetos de gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la definición de proceso. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia, utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

UserData (MQCHAR128)

`UserData` es una serie de caracteres que contiene información de usuario perteneciente a la aplicación que se va a iniciar. Esta información es para que la utilice una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio, o la aplicación iniciada por el supervisor desencadenante. La información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *`UserData`* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ pasa *`UserData`* a la aplicación iniciada como parte de la lista de parámetros. La lista de parámetros consta de la estructura MQTMC2 (que contiene *`UserData`*), seguida de un espacio en blanco, seguida de *`EnvData`* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario. Para Microsoft Windows, la serie de caracteres no debe contener comillas dobles si la definición de proceso se va a pasar a **`runmqtrm`**.

Para determinar el valor de este atributo, utilice el selector MQCA_USER_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_USER_DATA_LENGTH.

Códigos de retorno

Para cada llamada MQI (IBM MQ Message Queue Interface) y MQAI (IBM MQ Administration Interface), el gestor de colas o una rutina de salida devuelven un código de **terminación** y un código de **razón** para indicar el éxito o el error de la llamada.

Las aplicaciones no deben depender de los errores que se comprueban en un orden específico, excepto cuando se indique específicamente. Si puede surgir más de un código de terminación o código de razón de una llamada, el error concreto notificado depende de la implementación.

Las aplicaciones que comprueban la finalización correcta después de una llamada de API de IBM MQ siempre deben comprobar el código de finalización. No asuma el valor del código de terminación, basándose en el valor del código de razón.

Códigos de finalización

El parámetro de código de terminación (*`CompCode`*) permite al interlocutor ver rápidamente si la llamada se ha completado correctamente, se ha completado parcialmente o ha fallado.A continuación se muestra una lista de códigos de terminación, con más detalles de los que se proporcionan en las descripciones de las llamadas:

MQCC_OK
La llamada se ha completado del todo; se han establecido todos los parámetros de salida. En este caso, el parámetro **Reason** tiene siempre el valor MQRC_NONE.

MQCC_WARNING
La llamada se ha completado parcialmente. Es posible que algunos parámetros de salida se hayan establecido además de los parámetros de salida *`CompCode`* y *`Reason`* . El parámetro **Reason** proporciona información adicional sobre la finalización parcial.

MQCC_FAILED

El proceso de la llamada no se ha completado. El estado del gestor de colas no se modifica, excepto cuando se indique específicamente. Se han establecido los parámetros de salida *CompCode* y *Reason* ; otros parámetros no se modifican, excepto cuando se indique lo contrario.

La razón puede ser un error en el programa de aplicación, o puede ser el resultado de alguna situación externa al programa, por ejemplo, es posible que se haya revocado la autorización del usuario. El parámetro **Reason** proporciona información adicional sobre el error.

códigos de razón

El parámetro de código de razón (*Reason*) califica el parámetro de código de terminación (*CompCode*).

Si no hay que notificar ninguna razón especial, se devuelve MQRC_NONE. Una llamada que ha finalizado correctamente devuelve MQCC_OK y MQRC_NONE.

Si el código de terminación es MQCC_WARNING o MQCC_FAILED, el gestor de colas siempre informa de una razón calificadora; se proporcionan detalles en la descripción de cada llamada.

Cuando las rutinas de salida de usuario establecen códigos de terminación y razones, deben cumplir estas reglas. Además, cualquier valor de razón especial definido por las salidas de usuario debe ser menor que cero, para asegurarse de que no entra en conflicto con los valores definidos por el gestor de colas. Las salidas pueden establecer razones ya definidas por el gestor de colas, cuando proceda.

También aparecen códigos de razón en:

- El campo *Reason* de la estructura MQDLH.
- El campo *Feedback* de la estructura MQMD.

Para obtener descripciones completas de los códigos de razón, consulte Mensajes y códigos de razón.

Reglas para validar las opciones de MQI

Esta sección lista las situaciones que producen un código de razón MQRC_OPTIONS_ERROR a partir de una llamada MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE o MQSUB.

llamada MQOPEN

Para las opciones de la llamada MQOPEN:

- Se debe especificar al menos *uno* de los siguientes:
  - MQOO_BROWSE
  - MQOO_INPUT_EXCLUSIVE [1]
  - MQOO_INPUT_SHARED [1]
  - MQOO_INPUT_AS_Q_DEF [1]
  - MQOO_INQUIRE
  - MQOO_OUTPUT
  - MQOO_SET
  - MQOO_BIND_ON_OPEN [2]
  - MQOO_BIND_NOT_FIXED [2]
  - MQOO_BIND_ON_GROUP [2]
  - MQOO_BIND_AS_Q_DEF [2]
- Solo se permite *uno* de los siguientes:
  - MQOO_READ_AHEAD
  - MQOO_NO_READ_AHEAD
  - MQOO_READ_AHEAD_AS_Q_DEF

1. Solo se permite *uno* de los siguientes:
   - MQOO_INPUT_EXCLUSIVE
   - MQOO_INPUT_SHARED
   - MQOO_INPUT_AS_Q_DEF
2. Solo se permite *uno* de los siguientes:
   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NOT_FIXED
   - MQOO_BIND_ON_GROUP
   - MQOO_BIND_AS_Q_DEF

   Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQOO_BIND_AS_Q_DEF es cero, si se especifica con cualquiera de las otras dos opciones de vinculación no se genera el código de razón MQRC_OPTIONS_ERROR. MQOO_BIND_AS_Q_DEF se proporciona para ayudar a la documentación del programa.

- Si se especifica MQOO_SAVE_ALL_CONTEXT, también debe especificarse una de las opciones MQOO_INPUT_ *.
- Si se especifica una de las opciones MQOO_SET_ * _CONTEXT o MQOO_PASS_ * _CONTEXT, también debe especificarse MQOO_OUTPUT.
- Si se especifica MQOO_CO_OP, también se debe especificar MQOO_BROWSE
- Si se especifica MQOO_NO_MULTICAST, también debe especificarse MQOO_OUTPUT.

llamada MQPUT

Para las opciones de colocación de mensajes:

- La combinación de MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT no está permitida.
- Solo se permite *uno* de los siguientes:
  - MQPMO_DEFAULT_CONTEXT
  - MQPMO_NO_CONTEXT
  - MQPMO_PASS_ALL_CONTEXT
  - MQPMO_PASS_IDENTITY_CONTEXT
  - MQPMO_SET_ALL_CONTEXT
  - MQPMO_SET_IDENTITY_CONTEXT
- Solo se permite *uno* de los siguientes:
  - MQPMO_ASYNC_RESPONSE
  - MQPMO_SYNC_RESPONSE
  - MQPMO_RESPONSE_AS_TOPIC_DEF
  - MQPMO_RESPONSE_AS_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY no está permitido (sólo es válido en la llamada MQPUT1).

llamada MQPUT1

Para las opciones de transferencia de mensajes, las reglas son las mismas que para la llamada MQPUT, excepto para lo siguiente:

- Se permite MQPMO_ALTERNATE_USER_AUTHORITY.
- MQPMO_LOGICAL_ORDER no está permitido.

llamada MQGET

Para las opciones get-message:

- Solo se permite *uno* de los siguientes:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_SYNCPOINT
  - MQGMO_SYNCPOINT_IF_PERSISTENT
- Solo se permite *uno* de los siguientes:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT no está permitido con ninguno de los siguientes elementos:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_LOCK
  - MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT no está permitido con ninguno de los siguientes:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_COMPLETE_MSG
  - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT requiere que se especifique MQGMO_SYNCPOINT.
- La combinación de MQGMO_WAIT y MQGMO_SET_SIGNAL no está permitida.
- Si se especifica MQGMO_LOCK, también se debe especificar uno de los siguientes:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
- Si se especifica MQGMO_UNLOCK, sólo se permiten los valores siguientes:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_NO_WAIT

llamada MQCLOSE

Para las opciones de la llamada MQCLOSE:

- La combinación de MQCO_DELETE y MQCO_DELETE_PURGE no está permitida.
- Sólo se permite uno de los siguientes:
  - MQCO_KEEP_SUB
  - MQCO_REMOVE_SUB

Llamada MQSUB

Para las opciones de la llamada MQSUB:

- Se debe especificar al menos uno de los siguientes:
  - MQSO_ALTER
  - MQSO_RESUME
  - MQSO_CREATE
- Sólo se permite uno de los siguientes:
  - MQSO_DURABLE
  - MQSO_NON_DURABLE

  Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQSO_NON_DURABLE es cero, si se especifica con MQSO_DURABLE no se genera el código de razón MQRC_OPTIONS_ERROR. Se proporciona MQSO_NON_DURABLE para ayudar a la documentación del programa.
- La combinación de MQSO_GROUP_SUB y MQSO_MANAGED no está permitida.
- MQSO_GROUP_SUB requiere que se especifique MQSO_SET_CORREL_ID.
- Sólo se permite uno de los siguientes:
  - MQSO_ANY_USERID
  - MQSO_FIXED_USERID
- MQSO_NEW_PUBLICATIONS_ONLY está permitido en combinación con:
  - MQSO_CREATE
  - MQSO_ALTER, si se ha establecido MQSO_NEW_PUBLICATIONS_ONLY en la suscripción original
- La combinación de MQSO_PUBLICATIONS_ON_REQUEST y SubLevel mayor que 1 no está permitida.
- Sólo se permite uno de los siguientes:
  - MQSO_WILDCARD_CHAR
  - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST requiere que se especifique MQSO_MANAGED.

Mensajes del mandato de publicación/suscripción en cola

Una aplicación puede utilizar mensajes de mandato MQRFH2 para controlar una aplicación de publicación/suscripción en cola.

Una aplicación que utiliza MQRFH2 para la publicación/suscripción puede enviar los siguientes mensajes de mandato al SYSTEM.BROKER.CONTROL.QUEUE:

- “Mensaje de supresión de publicación” en la página 894
- “Mensaje de anulación de registro de suscriptor” en la página 895
- “Publicar mensaje” en la página 899
- “Mensaje Registrar suscriptor” en la página 901
- “Mensaje de solicitud de actualización” en la página 906

Si está escribiendo aplicaciones de publicación/suscripción en cola, debe comprender estos mensajes, el mensaje de respuesta del gestor de colas y el descriptor de mensaje (MQMD); consulte la información siguiente:

- “Mensaje de respuesta del gestor de colas” en la página 908
- “Valores de MQMD para publicaciones reenviadas por un gestor de colas” en la página 914
- “Valores de MQMD en mensajes de respuesta del gestor de colas” en la página 915
- “Códigos de razón de publicación/suscripción” en la página 910

Los mandatos están contenidos en una carpeta `psc` en el campo **`NameValueData`** de la cabecera `MQRFH2` . El mensaje que puede enviar un intermediario en respuesta a un mensaje de mandato está contenido en una carpeta `pscr` .

Las descripciones de cada mandato listan las propiedades que puede contener una carpeta. A menos que se especifique lo contrario, las propiedades son opcionales y sólo pueden aparecer una vez.

Los nombres de las propiedades se muestran como `<Command>`.

Los valores deben estar en formato de serie, por ejemplo: `Publish`.

Una constante de tipo serie que representa el valor de una propiedad se muestra entre paréntesis, por ejemplo: `(MQPSC_PUBLISH)`.

Las constantes de tipo serie se definen en el archivo de cabecera `cmqpsc.h` que se proporciona con el gestor de colas.

Mensaje de supresión de publicación

El mensaje del mandato **`Delete Publication`** se envía a un gestor de colas desde un publicador, o desde otro gestor de colas, para indicar al gestor de colas que suprima las publicaciones retenidas para los temas especificados.

Este mensaje se envía a una cola supervisada por la interfaz de publicación/suscripción en cola del gestor de colas.

La cola de entrada ha de ser la cola a la que se envió la publicación original.

Si tiene autorización para algunos, pero no para todos, los temas especificados en el mensaje de mandato **`Delete Publication`** , sólo se suprimirán esos temas. Un mensaje **`Broker Response`** indica qué temas no se suprimen.

De forma similar, si un mandato **`Publish`** contiene más de un tema, un mandato **`Delete Publication`** que coincide con algunos de estos temas, pero no todos, suprime sólo las publicaciones para los temas que se especifican en el mandato **`Delete Publication`** .

Consulte “Valores de MQMD para publicaciones reenviadas por un gestor de colas” en la página 914 para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Propiedades

Mandato (*MQPSC_COMMAND*)
El valor es `DeletePub` (*MQPSC_DELETE_PUBCIONTION*).

Esta propiedad ha de especificarse.

Tema > (*MQPSC_TOPIC*)
El valor es una serie de caracteres que contiene un tema para el cual han de suprimirse las publicaciones retenidas. Se pueden incluir caracteres comodín en la serie de caracteres para suprimir publicaciones sobre más de un tema.

Esta propiedad ha de especificarse; puede repetirse para tantos temas como se desee.

DelOpt (*MQPSC_DELETE_OPTION*)
La propiedad de opciones de supresión puede tener uno de los siguientes valores:

`Local` (*MQPSC_LOCAL*)

Todas las publicaciones retenidas para los temas especificados se suprimen en el gestor de colas local (es decir, el gestor de colas al que se envía este mensaje), tanto si se han publicado con la opción `Local` como si no.

Las publicaciones de otros gestores de colas no se ven afectadas.

None (*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que especificar la propiedad `DelOpt`. Si se especifican otras opciones simultáneamente, se ignora `None`.

El valor predeterminado si se omite esta propiedad es que todas las publicaciones retenidas para los temas especificados se suprimen en todos los gestores de colas de la red, independientemente de si se han publicado con la opción `Local` .

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **`Delete Publication`** . Lo utiliza la aplicación de ejemplo para suprimir, en el gestor de colas local, la publicación retenida que contiene la última puntuación en la coincidencia entre Team1 y Team2.

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

Mensaje de anulación de registro de suscriptor

El mensaje del mandato **`Deregister Subscriber`** lo envía a un gestor de colas un suscriptor, u otra aplicación en nombre de un suscriptor, para indicar que ya no desea recibir mensajes que coincidan con los parámetros especificados.

Este mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, la cola de control del gestor de colas. El usuario debe tener la autorización necesaria para colocar un mensaje en esta cola.

Consulte Valores de MQMD para publicaciones reenviadas por un gestor de colas para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Se puede anular el registro de una suscripción individual especificando el tema correspondiente, el punto de suscripción y los valores de filtro de la suscripción original. Si alguno de los valores no se ha especificado (es decir, han tomado los valores predeterminados) en la suscripción original, se deben omitir cuando se anule el registro de la suscripción.

Todas las suscripciones para un suscriptor, o un grupo de suscriptores, se pueden anular del registro utilizando la opción `DeregAll` . Por ejemplo, si se especifica `DeregAll` , junto con un punto de suscripción (pero sin tema ni filtro), se anularán el registro de todas las suscripciones para el suscriptor en el punto de suscripción especificado, independientemente del tema y del filtro. Se permite cualquier combinación de tema, filtro y punto de suscripción; si se especifican los tres, sólo puede coincidir una suscripción y se ignora la opción `DeregAll` .

El mensaje debe enviarlo el suscriptor que ha registrado la suscripción; esto se confirma comprobando el ID de usuario del suscriptor.

Un administrador del sistema también puede anular el registro de las suscripciones utilizando los mandatos MQSC o PCF. Sin embargo, las suscripciones registradas con una cola dinámica temporal están asociadas con la cola, no sólo con el nombre de cola. Si la cola se suprime, ya sea de forma explícita, o mediante la desconexión de la aplicación del gestor de colas, ya no es posible utilizar el mandato **`Deregister Subscriber`** para anular el registro de las suscripciones para dicha cola. Las suscripciones se pueden anular del registro utilizando el entorno de trabajo del desarrollador y el gestor de colas las elimina automáticamente la próxima vez que coincida con una publicación de la suscripción o la próxima vez que se reinicie el gestor de colas. En circunstancias normales, las aplicaciones deben anular el registro de sus suscripciones antes de suprimir la cola o de desconectarse del gestor de colas.

Si un suscriptor envía un mensaje para anular el registro de una suscripción y recibe un mensaje de respuesta para decir que se ha procesado correctamente, es posible que algunas publicaciones sigan llegando a la cola de suscriptores si el gestor de colas las estaba procesando al mismo tiempo que

se estaba anulando el registro de la suscripción. Si los mensajes no se eliminan de la cola, es posible que haya una acumulación de mensajes no procesados en la cola del suscriptor. Si la aplicación ejecuta un bucle que incluye una llamada MQGET con el `CorrelId` adecuado después de permanecer inactivo durante un tiempo, estos mensajes se borran de la cola.

De forma similar, si el suscriptor utiliza una cola dinámica permanente y anula el registro y cierra la cola con la opción *MQCO_DELETE_PURGE* en una llamada MQCLOSE, es posible que la cola no esté vacía. Si alguna publicación del gestor de colas todavía no se ha confirmado cuando se suprime la cola, la llamada MQCLOSE emite un código de retorno MQRC_Q_NOT_EMPTY. La aplicación puede evitar este problema durmiendo y volviendo a emitir la llamada MQCLOSE de vez en cuando.

Propiedades

Mandato (*MQPSC_COMMAND*)
El valor es `DeregSub` (*MQPSC_DEREGISTER_SUBSCRIBER*).

Esta propiedad ha de especificarse.

Tema (*MQPSC_TOPIC*)
El valor es una serie que contiene el tema que se va a anular el registro.

Esta propiedad, opcionalmente, se puede repetir si se van a anular el registro de varios temas. Se puede omitir si se especifica `DeregAll` en `<RegOpt>`.

Los temas especificados pueden ser un subconjunto de los que están registrados si el suscriptor desea retener suscripciones para otros temas. Se permiten caracteres comodín, pero una serie de tema que contiene caracteres comodín debe coincidir exactamente con la serie correspondiente que se ha especificado en el mensaje de mandato **`Deregister Subscriber`** .

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
El valor es una serie que especifica el punto de suscripción del que se va a desconectar la suscripción.

Esta propiedad no debe repetirse. Se puede omitir si se especifica un `< Topic>` , o si se especifica `DeregAll` en `<RegOpt>`. Si omite esta propiedad, se produce lo siguiente:

- Si **no** especifica `DeregAll`, las suscripciones que coincidan con la propiedad `< Topic>` (y la propiedad `< Filter >` , si está presente) se anularán el registro del punto de suscripción predeterminado.
- Si especifica `DeregAll`, todas las suscripciones (que coincidan con las propiedades `< Topic>` y `< Filter >` si están presentes) se anularán del registro de todos los puntos de suscripción.

Tenga en cuenta que no puede especificar el punto de suscripción predeterminado de forma explícita. Por lo tanto, no hay forma de anular el registro de todas las suscripciones de este punto de suscripción solamente; debe especificar los temas.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Se trata de una serie de longitud variable con una longitud máxima de 64 caracteres. Se utiliza para representar una aplicación con un interés en una suscripción. El gestor de colas mantiene un conjunto de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga una sola identidad o un número ilimitado de identidades.

Si `SubIdentity` está en el conjunto de identidades para la suscripción, se elimina del conjunto. Si el conjunto de identidades queda vacío como resultado de esto, la suscripción se elimina del gestor de colas, a menos que se especifique `LeaveOnly` como valor de la propiedad `RegOpt` . Si el conjunto de identidades sigue conteniendo otras identidades, la suscripción no se elimina del gestor de colas y el flujo de publicación no se interrumpe.

Si se especifica `SubIdentity` , pero la `SubIdentity` no está en el conjunto de identidades para la suscripción, el mandato **`Deregister Subscriber`** falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

Filtro (*MQPSC_FILTER*)

El valor es una serie que especifica el filtro que se va a anular el registro. Debe coincidir exactamente, incluyendo mayúsculas y minúsculas y cualquier espacio, con un filtro de suscripción que se haya registrado previamente.

Esta propiedad puede, opcionalmente, repetirse si se va a anular el registro de más de un filtro. Se puede omitir si se especifica un `< Topic>` , o si se especifica `DeregAll` en `<RegOpt>`.

Los filtros especificados pueden ser un subconjunto de los registrados si el suscriptor desea retener suscripciones para otros filtros.

RegOpt (*MQPSC_REGISTRATION_OPTION*)

La propiedad de opciones de registro puede tomar los valores siguientes:

`DeregAll`

(*MQPSC_DEREGISTER_ALL*)

Se anulará el registro de todas las suscripciones coincidentes registradas para este suscriptor.

Si especifica `DeregAll`:

- `< Topic>`, `<SubPoint>`y `< Filter >` se pueden omitir.
- `< Topic>` y `< Filtro >` se pueden repetir, si es necesario.
- `<SubPoint>` no debe repetirse.

Si **no** especifica `DeregAll`:

- Se debe especificar `< Topic>` y se puede repetir si es necesario.
- `<SubPoint>` y `< Filter >` se pueden omitir.
- `<SubPoint>` no debe repetirse.
- `< Filtro >` se puede repetir, si es necesario.

Si los temas y los filtros se repiten, se eliminarán todas las suscripciones que coincidan con todas las combinaciones de los dos. Por ejemplo, un mandato **`Deregister Subscriber`** que especifique tres temas y tres filtros intentará eliminar nueve suscripciones.

`CorrelAsId`

(*MQPSC_CORREL_ID_AS_IDENTITY*)

El `CorrelId` del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza para identificar al suscriptor. Debe coincidir con el `CorrelId` utilizado en la suscripción original.

`FullResp`

(*MQPSC_FULL_RESPONSE*)

Cuando se especifica `FullResp` , todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

Cuando se especifica `FullResp` , no se permite `DeregAll` en el mandato **`Deregister Subscriber`** . Tampoco es posible especificar varios temas. El mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*, en ambos casos.

`LeaveOnly`

(*MQPSC_LEAVE_ONLY*)

Cuando se especifica con una `SubIdentity` que está en el conjunto de identidades para la suscripción, la `SubIdentity` se elimina del conjunto de identidades para la suscripción. La suscripción no se elimina del gestor de colas, aunque el conjunto de identidades resultante esté vacío. Si el valor `SubIdentity` no está en el conjunto de identidades, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

Si se especifica `LeaveOnly` sin `SubIdentity`, el mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

Si no se especifica `LeaveOnly` ni una `SubIdentity` , la suscripción se elimina independientemente del contenido del conjunto de identidades para la suscripción.

NO
(MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad de opciones de registro. Si se especifican otras opciones simultáneamente, se ignora `None`.

`VariableUserId`
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Cualquier usuario puede modificar o anular el registro de la suscripción cuando tenga la autorización adecuada, evitando la comprobación existente de que el ID de usuario debe coincidir con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción que se va a anular el registro tiene establecido `VariableUserId` , se debe establecer al anular el registro para indicar qué suscripción se va a anular el registro. De lo contrario, el ID de usuario del mandato **`Deregister Subscriber`** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

QMgrName (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor. Debe coincidir con el `QMgrName` utilizado en la suscripción original.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMgr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

QName (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores. Debe coincidir con el `QName` utilizado en la suscripción original.

Si se omite esta propiedad, el valor predeterminado es el nombre `ReplyToQ` en el descriptor de mensaje (MQMD), que no debe estar en blanco.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Si especifica SubName en un mandato **`Deregister Subscriber`** , el valor SubName tiene prioridad sobre todos los demás campos de identificador excepto el ID de usuario, a menos que `VariableUserId` esté establecido en la propia suscripción. Si no se ha establecido `VariableUserId` , el mandato **`Deregister Subscriber`** sólo se ejecuta correctamente si el ID de usuario del mensaje de mandato coincide con el de la suscripción, si no es así, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si existe una suscripción que coincide con la identidad tradicional de este mandato pero no tiene ningún SubName , el mandato **`Deregister Subscriber`** falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*. Si se intenta anular el registro de una suscripción que tiene un SubName utilizando un mensaje de mandato que coincide con la identidad tradicional pero sin ningún SubName especificado, el mandato se ejecuta correctamente.

SubUser(*MQPSC_SUBSCRIPTION_USER_DATA*)
Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de la publicación al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo es para el uso de la aplicación.

SubUserLos datos se devuelven en la información metatópica (MQCACF_REG_SUB_USER_DATA) para una suscripción, si SubUserData está presente.

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **`Deregister Subscriber`** . En este ejemplo, la aplicación de ejemplo anula el registro de su suscripción a los temas que contienen la última puntuación para todas las coincidencias. La identidad del suscriptor, incluido el `CorrelId`, se toma de los valores predeterminados en MQMD.

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publicar mensaje

El mensaje del mandato **`Publish`** se coloca en una cola, o desde un gestor de colas a un suscriptor, para publicar información sobre un tema o temas especificados.

Es necesaria la autorización para colocar un mensaje en una cola y la autorización para publicar información sobre un tema o temas especificados.

Si el usuario tiene autorización para publicar información sobre algunos temas, pero no todos, sólo se utilizan estos temas para publicar; una respuesta de aviso indica qué temas no se utilizan para publicar.

Si un suscriptor tiene suscripciones coincidentes, el gestor de colas reenvía el mensaje **`Publish`** a las colas de suscriptor definidas en los mensajes de mandato **`Register Subscriber`** correspondientes.

Consulte Mensaje de respuesta del gestor de colas para obtener detalles de los parámetros del descriptor de mensaje (MQMD) necesarios al enviar un mensaje de mandato al gestor de colas y utilizados cuando un gestor de colas reenvía una publicación a un suscriptor.

El gestor de colas reenvía el mensaje **`Publish`** a otros gestores de colas de la red que tienen suscripciones coincidentes, a menos que sea una publicación local.

Los datos de la publicación, si los hay, se incluyen en el cuerpo del mensaje. Los datos se pueden describir en una carpeta `<mcd>`, en el campo `NameValueData` de la cabecera MQRFH2.

Propiedades

Mandato (*MQPSC_COMMAND*)
El valor es `Publish` (*MQPSC_PUBLISH*).

Esta propiedad ha de especificarse.

Tema (*MQPSC_TOPIC*)
El valor es una serie de caracteres que contiene un tema que clasifica esta publicación. No se permiten caracteres comodín.

Debe añadir el tema a la lista de nombres SYSTEM.QPUBSUB.QUEUE.NAMELIST, consulte Adición de una corriente para obtener instrucciones sobre cómo completar esta tarea.

Esta propiedad ha de especificarse y puede repetirse, opcionalmente, para tantos temas como se desee.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Es el punto de suscripción en el que se publica la publicación.

En WebSphere Event Broker 6.0, el valor de la propiedad <SubPoint> es el valor del atributo Punto de suscripción del nodo Publication que maneja la publicación.

En IBM WebSphere MQ 7.0.1, el valor de la propiedad <SubPoint> debe coincidir con el nombre de un punto de suscripción. Consulte Adición de un punto de suscripción.

PubOpt (*MQPSC_PUBCIONTION_OPTION*)
La propiedad de opciones de publicación puede tener uno de los siguientes valores:

RetainPub
(*MQPSC_RETAIN_PUB*)

El gestor de colas debe retener una copia de la publicación. Si esta opción no está establecida, la publicación se suprime tan pronto como el gestor de colas ha enviado la publicación a todos sus suscriptores actuales.

IsRetainedPub
(*MQPSC_IS_RETAINED_PUB*)

(Sólo puede ser establecido por un gestor de colas.) Esta publicación ha sido retenida por el gestor de colas. El gestor de colas establece esta opción para notificar a un suscriptor que esta publicación se ha publicado anteriormente y se ha retenido, siempre que la suscripción se haya registrado con la opción `InformIfRetenido` . Se establece únicamente en respuesta a un mensaje de mandato de `registro de suscriptor` o de `petición de actualización`. Las publicaciones retenidas que se envían directamente a suscriptores no tiene establecida esta opción.

Local
(*MQPSC_LOCAL*)

Esta opción indica al gestor de colas que esta publicación no debe enviarse a otros gestores de colas. Todos los suscriptores registrados en este gestor de colas reciben esta publicación si tienen suscripciones coincidentes.

OtherSubsOnly
(*MQPSC_OTHER_SUBS_ONLY*)

Esta opción permite el proceso más sencillo de aplicaciones de tipo conferencia, en las que un publicador es también suscriptor del mismo tema. Indica al gestor de colas que no envíe la publicación a la cola de suscriptores del publicador aunque tenga una suscripción coincidente. La cola de suscriptores del publicador consta de su `QMgrName`, `QName`y `CorrelId`opcional, tal como se describe en la lista siguiente.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El `CorrelId` del MQMD (que no ha de ser cero) forma parte de la cola de suscriptores del publicador en aplicaciones donde el publicador es también un suscriptor.

Ninguno
(*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto surte el mismo efecto que omitir la propiedad de opciones de publicación. Si se especifican otras opciones simultáneamente, se ignora None.

Para tener varias opciones de publicación, solo es necesario introducir elementos <PubOpt> adicionales.

El valor predeterminado, si se omite esta propiedad, es que no se establece ninguna opción de publicación.

PubTime (*MQPSC_PUBLISH_TIMESTAMP*)
El valor es una indicación de la hora de publicación, opcional, establecida por el publicador. Tiene 16 caracteres de largo con el formato:

```
YYYYMMDDHHMMSSTH
```

y utiliza la Hora Universal. El gestor de colas no comprueba esta información antes de enviarla a los suscriptores.

SeqNum (*MQPSC_SEQUENCE_NUMBER*)
El valor es un número de secuencia opcional establecido por el publicador.

Debe incrementarse en 1 con cada publicación. Sin embargo, esto no lo comprueba el gestor de colas, que simplemente transmite esta información a los suscriptores.

Si las publicaciones sobre el mismo tema se publican en distintos gestores de colas interconectados, es responsabilidad de los publicadores asegurarse de que los números de secuencia, si se utilizan, sean significativos.

QMgrName (*MQPSC_Q_MGR_NAME*)
El valor es una serie que contiene el nombre del gestor de colas para la cola de suscriptores del publicador, en aplicaciones en las que el publicador también es un suscriptor (consulte `SóloOtherSubs`).

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMgr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

QName (*MQPSC_Q_NAME*)
El valor es una serie que contiene el nombre de la cola de suscriptores del publicador, en aplicaciones en las que el publicador también es un suscriptor (consulte `OtherSubs`).

Si se omite esta propiedad, el valor predeterminado es el nombre de la cola de respuestas (`ReplyToQ`) en el descriptor del mensaje (MQMD), que no ha de estar en blanco si se ha establecido `OtherSubsOnly`.

Ejemplo

A continuación se muestran algunos ejemplos de *NameValueData* para un mensaje de mandato **`Publish`** .

El primer ejemplo es para una publicación enviada por el simulador de coincidencias en la aplicación de ejemplo para indicar que se ha iniciado una coincidencia.

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

El segundo ejemplo es para una publicación retenida. Se publica la última puntuación entre Team1 y Team2.

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Mensaje Registrar suscriptor

El mensaje del mandato **`Register Subscriber`** lo envía a un gestor de colas un suscriptor, o otra aplicación en nombre de un suscriptor, para indicar que desea suscribirse a uno o más temas en un punto de suscripción. También se puede especificar un filtro de contenido de mensaje.

En las expresiones de filtro de publicación/suscripción, el anidamiento de paréntesis hace que el rendimiento disminuya exponencialmente. Evite anidar paréntesis a una profundidad mayor que aproximadamente 6.

El mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, que es la cola de control del gestor de colas. Se necesita autorización para transferir un mensaje a esta cola, además de la autorización de acceso (establecida por el administrador del sistema del gestor de colas) para el tema o temas de la suscripción.

Si el usuario tiene autorización sobre algunos temas, pero no todos, sólo se registran los que tienen autorización; una respuesta de aviso indica los que no están registrados.

Consulte "Valores de MQMD en los mensajes de mandato para el gestor de colas" en la página 913 para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Si la respuesta a la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

Propiedades

Mandato (*MQPSC_COMMAND*)
El valor es RegSub (*MQPSC_REGISTER_SUBSCRIBER*). Esta propiedad ha de especificarse.

Tema (*MQPSC_TOPIC*)
El tema para el que el suscriptor desea recibir publicaciones. Los caracteres comodín se pueden especificar como parte del tema.

Si utiliza el mandato MQSC **`display sub`** para examinar la suscripción creada de esta forma, el valor de la etiqueta < `Topic`> se muestra como la propiedad TOPICSTR de la suscripción.

Esta propiedad es necesaria y, opcionalmente, se puede repetir para tantos temas como sea necesario.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

En WebSphere Event Broker 6.0, el valor de la propiedad <`SubPoint`> debe coincidir con el valor del atributo Punto de suscripción de los nodos de publicación a los que están suscritos.

En IBM WebSphere MQ 7.0.1, el valor de la propiedad <`SubPoint`> debe coincidir con el nombre de un punto de suscripción. Consulte Adición de un punto de suscripción.

Filtro (*MQPSC_FILTER*)
El valor es una expresión SQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor. Esta propiedad corresponde a la serie de selección que se utiliza en las llamadas MQSUB y MQOPEN. Para obtener más información, consulte Selección del contenido de un mensaje

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
Esta propiedad Opciones de registro puede tomar los valores siguientes:

`AddName`
(*MQPSC_ADD_NAME*)

Cuando se especifica para una suscripción existente que coincide con la identidad tradicional de este mandato Registrar suscripción, pero sin ningún valor SubName actual, el SubName especificado en este mandato se añade a la suscripción.

Si se especifica AddName , el campo SubName es obligatorio; de lo contrario, se devuelve MQRCCF_REG_OPTIONS_ERROR.

`CorrelAsId`
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El `CorrelId` del descriptor de mensaje (MQMD) se utiliza al enviar publicaciones coincidentes a la cola de suscriptores. El `CorrelId` no debe ser cero,

FullResp
(*MQPSC_FULL_RESPONSE*)

Cuando se especifica, todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

FullResp sólo es válido cuando el mensaje de mandato hace referencia a una única suscripción. Por lo tanto, sólo se permite un tema en el mandato; de lo contrario, el mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

InformIfRet
(*MQPSC_INFORM_IF_RETAINED*)

El gestor de colas informa al suscriptor si una publicación se retiene cuando envía un mensaje de publicación en respuesta a un mensaje de mandato **`Register Subscriber`** o **`Request Update`** . El gestor de colas lo hace incluyendo la opción de publicación `IsRetainedPub` en el mensaje.

JoinExcl
(*MQPSC_JOIN_EXCLUSIVE*)

Esta opción indica que la `SubIdentity` especificada se debe añadir como miembro exclusivo del conjunto de identidades para la suscripción y que no se pueden añadir otras identidades al conjunto.

Si la identidad ya se ha unido a 'shared' y es la única entrada del conjunto, el conjunto se cambia a un bloqueo exclusivo mantenido por esta identidad. De lo contrario, si la suscripción tiene actualmente otras identidades en el conjunto de identidades (con acceso compartido), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*.

JoinShared
(*MQPSC_JOIN_SHARED*)

Esta opción indica que la `SubIdentity` especificada debe añadirse al conjunto de identidades para la suscripción.

Si la suscripción está bloqueada actualmente de forma exclusiva (utilizando la opción `JoinExcl`), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_LOCKED*, a menos que la identidad que tiene la suscripción bloqueada sea la misma que la de este mensaje de mandato. En este caso, el bloqueo se modifica automáticamente a un bloqueo compartido.

Local
(*MQPSC_LOCAL*)

La suscripción es local y no se distribuye a otros gestores de colas de la red. Las publicaciones realizadas en otros gestores de colas no se entregan a este suscriptor, a menos que también tenga una suscripción global correspondiente.

SóloNewPubs
(*MQPSC_NEW_PUBS_ONLY*)

Las publicaciones retenidas que existen en el momento en que se registra la suscripción no se envían al suscriptor; sólo se envían nuevas publicaciones.

Si un suscriptor se vuelve a registrar y cambia esta opción para que ya no se establezca, es posible que se vuelva a enviar una publicación que ya se le ha enviado.

NoAlter
(*MQPSC_NO_ALTER*)

Los atributos de una suscripción coincidente existente no se cambian.

Cuando se está creando una suscripción, esta opción se ignora. Todas las demás opciones especificadas se aplican a la nueva suscripción.

Si una `SubIdentity` también tiene una de las opciones de unión (`JoinExcl` o `JoinShared`) , la identidad se añade al conjunto de identidades independientemente de si se especifica `NoAlter` .

Ninguno
(*MQPSC_NONE*)

Todas las opciones de registro toman sus valores predeterminados.

Si el suscriptor ya está registrado, sus opciones se restablecen a sus valores predeterminados (tenga en cuenta que esto no tiene el mismo efecto que omitir la propiedad de opciones de registro) y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber** .

Si se especifican otras opciones de registro al mismo tiempo, se ignora `Ninguna` .

NonPers
(*MQPSC_NON_PERSISTENT*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes no persistentes.

Pers
(*MQPSC_PERSISTENT*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes persistentes.

PersAsPub
(*MQPSC_PERSISTENT_AS_PUBLISH*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada por el publicador. Éste es el comportamiento predeterminado.

Cola dePersAs
(*MQPSC_PERSISTENT_AS_Q*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada en la cola de suscriptor.

PubOnReqOnly
(*MQPSC_PUB_ON_REQUEST_ONLY*)

El gestor de colas no envía publicaciones al suscriptor, excepto en respuesta a un mensaje de mandato **Request Update** .

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Esto permite a cualquier usuario modificar o anular el registro de la suscripción si el usuario tiene la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción del mandato **Request Update** tiene establecido `VariableUserId` , se debe establecer en el momento de la solicitud de actualización para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **Request Update** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

Si un mensaje de mandato **Register Subscriber** sin este conjunto de opciones hace referencia a una suscripción existente que tiene este conjunto de opciones, la opción se elimina de esta suscripción y el ID de usuario de la suscripción se arregla ahora. Si ya existe un suscriptor que tiene la misma identidad (cola, gestor de colas e identificador de correlación) pero con un ID de usuario diferente asociado, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY* porque sólo puede haber un ID de usuario asociado a una identidad de suscriptor.

Si se omite la propiedad de opciones de registro y el suscriptor ya está registrado, sus opciones de registro no se modifican y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber** .

Si el suscriptor todavía no está registrado, se crea una nueva suscripción con todas las opciones de registro que toman sus valores predeterminados.

Los valores predeterminados son PersAsPub y no se ha establecido ninguna otra opción.

QMgrName (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es QMgrNamedel gestor de colas.

QName (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor predeterminado es el nombre ReplyToQ en el descriptor de mensaje (MQMD), que no debe estar en blanco en este caso.

Si la cola es una cola dinámica temporal, entrega no persistente de publicaciones (NonPers) debe especificarse en la propiedad <RegOpt> .

Si la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Se trata de un nombre asignado a una suscripción determinada. Puede utilizarlo en lugar del gestor de colas, cola y correlId opcional para hacer referencia a una suscripción.

Si ya existe una suscripción con este **SubName** , cualquier otro atributo de la suscripción (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData y Caducidad) se altera temporalmente con los atributos, si se especifican, que se pasan en el nuevo mensaje del mandato Registrar suscriptor . Sin embargo, si se utiliza **SubName** sin ningún campo QName especificado y se especifica una cola ReplyToen la cabecera MQMD, la cola de suscriptor cambia a ReplyToQ.

Si ya existe una suscripción que coincide con la identidad tradicional de este mandato, pero no tiene ningún **SubName** , el mandato Registration falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*, a menos que se especifique la opción **AddName** .

Si intenta modificar una suscripción con nombre existente utilizando otro mandato Registrar suscriptor que especifique el mismo **SubName** , y los valores de Topic, QMgrName, QName y CorrelId en el nuevo mandato coinciden con una suscripción existente diferente, con o sin un SubName definido, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*. Esto impide que dos nombres de suscripción hagan referencia a la misma suscripción.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Esta serie se utiliza para representar una aplicación con un interés en una suscripción. Es una serie de caracteres de longitud variable con una longitud máxima de 64 caracteres y es opcional. El gestor de colas mantiene un conjunto de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga sólo una identidad o un número ilimitado de identidades (consulte las opciones **JoinShared** y **JoinExcl**).

Un mandato subscribe que especifica la opción **JoinShared** o **JoinExcl** añade la **SubIdentity** al conjunto de identidades de la suscripción, si todavía no está presente y si el conjunto de identidades existente permite dicha acción; es decir, ningún otro suscriptor se ha unido de forma exclusiva o el conjunto de identidades está vacío.

Cualquier modificación de los atributos de la suscripción como resultado de un mandato `Registrar suscriptor` en el que se especifica una **SubIdentity** , sólo se ejecuta correctamente si sería el único miembro del conjunto de identidades para esta suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*. Esto impide que los atributos de una suscripción cambien sin que otros suscriptores interesados sean conscientes.

Si especifica una serie de caracteres de más de 64 caracteres, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

SubUser(*MQPSC_SUBSCRIPTION_USER_DATA*)
Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de publicaciones al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo está ahí para el uso de la aplicación.

Los datos de **SubUser** se devuelven en la información metatópica (*MQCACF_REG_SUB_USER_DATA*) para una suscripción si está presente.

Si especifica más de uno de los valores de opción de registro `NonPers, PersAsPub, PersAsQueue, and Pers`, sólo se utiliza el último. No puede combinar estas opciones en una suscripción individual.

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **Register Subscriber** . En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para registrar una suscripción a los temas que contienen las últimas puntuaciones en todas las coincidencias, con la opción 'Persistente como publicación' establecida. La identidad del suscriptor, incluido el `CorrelId`, se toma de los valores predeterminados en MQMD.

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de solicitud de actualización

El mensaje del mandato **Request Update** se envía desde un suscriptor a un gestor de colas, para solicitar las publicaciones retenidas actuales para el tema especificado y el punto de suscripción que coinciden con el filtro especificado (opcional).

Este mensaje se envía a *SYSTEM.BROKER.CONTROL.QUEUE*, la cola de control del gestor de colas. Se necesita autorización para colocar un mensaje en esta cola, además de autorización de acceso para el tema en la solicitud de actualización; esto lo establece el administrador del sistema del gestor de colas.

Este mandato se utiliza normalmente si el suscriptor ha especificado la opción `PubOnReqOnly` cuando se ha registrado. Si el gestor de colas tiene publicaciones retenidas coincidentes, se envían al suscriptor. Si el gestor de colas no tiene publicaciones retenidas coincidentes, la solicitud falla con el código de retorno *MQRCCF_NO_RETAINED_MSG*. El solicitante debe haber registrado previamente una suscripción con los mismos valores de Tema, SubPointy Filtro.

Propiedades

Mandato (*MQPSC_COMMAND*)
El valor es `ReqUpdate` (*MQPSC_REQUEST_UPDATE*). Esta propiedad ha de especificarse.

Tema (*MQPSC_TOPIC*)
El valor es el tema que solicita el suscriptor; se permiten caracteres comodín.

Esta propiedad debe especificarse, pero sólo se permite una aparición en este mensaje.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

Filtro (*MQPSC_FILTER*)
El valor es una expresión ESQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor.

La propiedad < `Filtro` > debe tener el mismo valor que el especificado en la suscripción original para la que está solicitando una actualización.

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
La propiedad de opciones de registro puede tomar el valor siguiente:

`CorrelAsId`
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El `CorrelId` del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza al enviar publicaciones coincidentes a la cola de suscriptores.

NO
(*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad `<RegOpt>` . Si se especifican otras opciones simultáneamente, se ignora `None`.

`VariableUserId`
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Esto permite a cualquier usuario modificar o anular el registro de la suscripción cuando tenga la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la suscripción original.

Si la suscripción del mandato **`Request Update`** tiene establecido `VariableUserId` , se debe establecer en el momento de la solicitud de actualización para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **`Request Update`** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

QMgrName (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMgr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es `QMgrName`del gestor de colas.

QName (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor predeterminado es el nombre `ReplyToQ` en el descriptor de mensaje (MQMD), que no debe estar en blanco en este caso.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Se trata de un nombre asignado a una suscripción determinada. Si se especifica en un mandato **`Request Update`** , el valor `SubName` tiene prioridad sobre todos los demás campos de identificador excepto el ID de usuario, a menos que `VariableUserId` esté establecido en la propia suscripción. Si no se ha establecido `VariableUserId` , el mandato *Request Update* sólo se ejecuta correctamente si el ID de usuario del mensaje de mandato coincide con el de la suscripción. Si el ID de usuario del mensaje de mandato no coincide con el de la suscripción, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si se ha establecido `VariableUserId` , y el ID de usuario difiere del de la suscripción, el mandato se ejecuta correctamente si el ID de usuario del nuevo mensaje de mandato tiene autorización para examinar la cola de secuencia y colocarla en la cola de suscriptor de la suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_NOT_AUTHORIZED*.

Si existe una suscripción que coincide con la identidad tradicional de este mandato, pero no tiene ningún `SubName`, el mandato **`Request Update`** falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*.

Si se intenta solicitar una actualización para una suscripción que tiene un `SubName` utilizando un mensaje de mandato que coincide con la identidad tradicional, pero sin ningún `SubName` especificado, el mandato se ejecuta correctamente.

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **`Request Update`** . En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para solicitar publicaciones retenidas que contienen las puntuaciones más recientes para todos los equipos. La identidad del suscriptor, incluido el `CorrelId`, se toma de los valores predeterminados en MQMD.

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de respuesta del gestor de colas

Un mensaje **`Queue Manager Response`** se envía desde un gestor de colas a la cola `ReplyTo` de un publicador o un suscriptor, para indicar el éxito o el error de un mensaje de mandato recibido por el gestor de colas si el descriptor de mensaje de mandato especifica que se necesita una respuesta.

El mensaje de respuesta está contenido en el campo `NameValueData` de la cabecera MQRFH2, en una carpeta `<pscr>`.

En el caso de un aviso o error, el mensaje de respuesta contiene la carpeta `<psc>` del mensaje de mandato, así como la carpeta `<pscr>`. Los datos del mensaje, si los hay, no están contenidos en el mensaje de respuesta del gestor de colas. Si se trata de un error, ninguno de los mensajes que haya causado el error se procesará; si se trata de un aviso, es posible que alguno de los mensajes se haya procesado correctamente.

Si se produce una anomalía al enviar una respuesta:

- Para los mensajes de publicación, el gestor de colas intenta enviar la respuesta a la cola de mensajes no entregados IBM MQ si la MQPUT falla. Esto permite enviar la publicación a suscriptores incluso si la respuesta no puede devolver al publicador.

- Para otros mensajes o si la respuesta a la publicación no puede enviarse a la cola de mensajes no entregados, se anota un error y, normalmente, el mensaje se sustituye. El comportamiento depende de cómo se haya configurado el nodo MQInput.

Propiedades

Finalización (*MQPSCR_COMPLETE*)
Es el código de terminación, que puede tener uno de estos tres valores.

`Aceptar`
El mandato ha terminado correctamente

`warning`
El mandato ha terminado pero con un error

`Error`
El comando ha fallado

Respuesta (*MQPSCR_RESPONSE*)
La respuesta a un mensaje de mandato, si dicho mandato ha devuelto un código de terminación `aviso` o `error`. Contiene una propiedad `<Reason>` y puede contener otras propiedades que indiquen la causa del aviso o error.

Si hay uno o más errores, habrá sólo una carpeta de respuesta que indicará únicamente la causa del primer error. En el caso de uno o más avisos, habrá una carpeta de respuesta para cada aviso.

Razón (*MQPSCR_REASON*)
Es el código de razón que califica al código de terminación, si el código de terminación es un `aviso` o `error`. Se establece en uno de los códigos de error listados en el ejemplo siguiente. La propiedad `<Reason>` está contenida en una carpeta `<Response>`. El código de razón puede ir seguido de cualquier propiedad válida de la carpeta `<psc>` (por ejemplo, un nombre de tema), que indique la causa del error o aviso. Si obtiene un código de razón de? ???, comprobar la exactitud de los datos, por ejemplo, corchetes coincidentes (< >).

Ejemplos

A continuación se muestran algunos ejemplos de `NameValueData` en un mensaje **`Queue Manager Response`** . Una respuesta satisfactoria podría ser la siguiente:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Este es un ejemplo de respuesta de anomalía; la anomalía es un error de filtro. La primera serie `NameValueData` contiene la respuesta; la segunda contiene el mandato original.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Este es un ejemplo de respuesta de aviso (debida a temas no autorizados). La primera serie `NameValueData` contiene la respuesta; la segunda serie `NameValueData` contiene el mandato original.

```
<pscr>
  <Completion>warning</Completion>
  <Response>
```

```
      <Reason>3081</Reason>
      <Topic>topic1</Topic>
    </Reponse>
    <Response>
      <Reason>3081</Reason>
      <Topic>topic2</Topic>
    </Reponse>
  </pscr>

  <psc>
   ...
   command message (to which
   the queue manager is responding)
   ...
  </psc>
```

Códigos de razón de publicación/suscripción

Estos códigos de razón se pueden devolver en el campo Razón de una carpeta <pscr> de respuesta de publicación/suscripción. También se listan las constantes que se pueden utilizar para representar estos códigos en los lenguajes de programación C o C++.

Las constantes MQRC_ requieren el archivo de cabecera IBM MQ cmqc.h . Las constantes MQRCCF_ requieren el archivo de cabecera IBM MQ cmqcfc.h (aparte de *MQRCCF_FILTER_ERROR* y *MQRCCF_INJU_USER*, que requieren el archivo de cabecera cmqpsc.h).

Código de razón y texto	Explicación	Emitido por
2336 MQRC_RFH_COMMAND_ERROR	Los valores válidos para el campo < Command> de una carpeta <psc> son: RegSub, DeregSub, Publish, DeletePuby ReqUpdate. Cualquier otro valor hace que se emita este código de error.	Cualquier mandato
2337 MQRC_RFH_ PARM_ERROR	Las carpetas <psc> y <mcd> tienen un conjunto de parámetros válidos que se pueden especificar dentro de ellas. Compruebe las descripciones de estas carpetas y asegúrese de que no ha especificado parámetros incorrectos.	Cualquier mandato
2338 MQRC_RFH_DUPLICATE_PARM	Algunos parámetros (por ejemplo, Tema) dentro de una carpeta <psc> se pueden repetir, pero otros (por ejemplo, Mandato) no se pueden repetir. Compruebe que no ha duplicado un parámetro no repetible.	Cualquier mandato
2339 MQRC_RFH_PARM_MISSING	Algunos parámetros de las carpetas <psc> o <mcd> son opcionales y se pueden omitir; algunos son obligatorios y no se deben omitir. Compruebe que ha incluido todos los parámetros obligatorios en las carpetas <psc> y <mcd> .	Cualquier mandato

Código de razón y texto	Explicación	Emitido por
2551 MQRC_SELECTION_NOT_AVAILABLE	No había ningún proveedor de selección de mensajes ampliado disponible para determinar qué suscriptores con un filtro especificado deben recibir la publicación.	Publicar, registrar suscriptor y solicitar actualización
	No había ningún proveedor de selección de mensajes ampliado disponible para manejar el filtro del suscriptor especificado.	Registrar suscriptor y solicitar actualización
2554 ERROR DE MQRC_CONTENT_	Un proveedor de selección de mensajes ampliados ha encontrado un error en la publicación actual o retenida.	Publicar y solicitar actualización
3008 MQRCCF_COMMAND_FAILED	Se ha producido un error interno que ha impedido que el mandato se ejecutara correctamente. El error puede producirse si se vuelve a emitir el mandato. El registro de sucesos del sistema para el gestor de colas contiene información que se debe utilizar al notificar el problema a IBM.	Cualquier mandato
3072 MQRCCF_TOPIC_ERROR	Uno o varios de los valores que ha proporcionado para el parámetro Topic son incorrectos. Compruebe que los valores de Topic se ajustan a las restricciones especificadas.	Cualquier mandato
3073 MQRCCF_NO_REGISTRADO	La combinación de SubPoint, Topic y Filter que ha especificado en el mandato DeregSub o ReqUpdate no era una combinación con la que se había registrado anteriormente o, para el mandato DeregSub si se ha especificado la opción DeregAll , no se ha utilizado una de las propiedades SubPoint, Topic o Filter para anular el registro de ninguna suscripción.	Mandatos de anulación de registro de suscriptor y solicitud de actualización
3074 MQRCCF_Q_MGR_NAME_ERROR	El gestor de colas especificado no era válido, o el gestor de colas no estaba disponible o no existía.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3076 MQRCCF_Q_NAME_ERROR	El nombre de cola especificado no era válido o la cola no existía en el gestor de colas especificado.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3077 MQRCCF_NO_RETAINED_MSG	No había mensajes retenidos para el tema que ha especificado. Esto puede ser o no un error, en función del diseño del programa de aplicación.	Mandato de solicitud de actualización

Código de razón y texto	Explicación	Emitido por
3079 MQRCCF_INCORRECT_Q	Los mandatos RegSub, DeregSuby ReqUpdate siempre se envían al SYSTEM.BROKER.CONTROL.QUEUE del gestor de colas para el que están destinados. Los mandatos de publicación y supresión se envían a la cola de entrada para el flujo de mensajes de publicación/suscripción concreto para el que están previstos; esto se determina cuando se diseña el flujo de mensajes. Este código de error se devuelve si se envía un mandato a la cola incorrecta.	Cualquier mandato
3080 MQRCCF_ID_CORREL_ERROR	Ha especificado CorrelAsId como uno de los parámetros RegOpt . Sin embargo, el campo CorrelId del MQMD no contiene un identificador de correlación válido (es decir, se establece en MQCI_NONE).	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3081 MQRCCF_NO_AUTORIZADO	No tiene autorización para realizar la acción solicitada. Los valores de autorización para el gestor de colas los maneja el administrador del sistema utilizando el editor Jerarquía de temas.	Publicar y registrar mandatos de suscriptor
3083 MQRCCF_REG_OPTIONS_ERROR	Ha especificado un parámetro RegOpt no reconocido en la carpeta <psc> que contiene el mandato RegSub o DeregSub .	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3084 MQRCCF_PUB_OPTIONS_ERROR	Ha especificado un parámetro PubOpt no reconocido en la carpeta <psc> que contiene el mandato Publish.	Mandato de publicación
3087 MQRCCF_DEL_OPTIONS_ERROR	Ha especificado un parámetro DelOpt no reconocido en la carpeta <psc> que contiene el mandato DeletePub .	Mandato Suprimir publicación
3150 MQRCCF_FILTER_ERROR	El valor especificado para el parámetro Filtro no es válido. Compruebe la sección que describe la sintaxis válida para las expresiones de filtro y asegúrese de que la expresión se ajusta.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3151 MQRCCF_USUARIO_INCORRECTO	Ya existe una suscripción que coincide con la especificada; sin embargo, la ha registrado un usuario diferente. Una suscripción sólo puede ser cambiada o anulada del registro por el usuario que la registró originalmente.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Ya existe una suscripción coincidente con un nombre de suscripción diferente.	

Código de razón y texto	Explicación	Emitido por
3153 MQRCCF_SUB_NAME_ERROR	El formato del nombre de suscripción no es válido o ya existe una suscripción coincidente sin nombre de suscripción.	
3154 MQRCCF_SUB_IDENTITY_ERROR	El parámetro de identidad de suscripción contiene un error. El valor proporcionado supera la longitud máxima permitida, o la identidad de suscripción no es actualmente miembro del conjunto de identidades de la suscripción y no se ha especificado una opción de registro de unión.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Un miembro del conjunto de identidades ha intentado modificar o anular el registro de una suscripción cuando no era el único miembro de este conjunto.	
3156 MQRCCF_SUBSCRIPTION_LOCKED	La suscripción está actualmente bloqueada de modo exclusivo por otra identidad.	
3157 MQRCCF_ALREADY_JOINED	Se ha especificado una opción de registro de unión, pero la identidad del suscriptor ya era miembro del conjunto de identidades de la suscripción.	

Valores de MQMD en los mensajes de mandato para el gestor de colas

Las aplicaciones que envían mensajes de mandatos al gestor de colas utilizan los siguientes valores de campos en el descriptor de mensaje (MQMD). Los campos que se dejan como valor predeterminado, o que se pueden establecer en cualquier valor válido de la forma habitual, no se listan aquí.

`Report`
Consulte `MsgType` y `CorrelId`.

`MsgType`
`MsgType` debe establecerse en *MQMT_REQUEST* o *MQMT_DATAGRAM*. Se devolverá *MQRC_MSG_TYPE_ERROR* si `MsgType` no está establecido en uno de estos valores.

`MsgType` debe establecerse en *MQMT_REQUEST* para un mensaje de mandato si siempre se necesita una respuesta. Los distintivos MQRO_PAN y MQRO_NAN del campo `Informe` no son significativos en este caso.

Si `MsgType` se establece en *MQMT_DATAGRAM*, las respuestas dependen del valor de los distintivos MQRO_PAN y MQRO_NAN en el campo `Informe` :

- Sólo MQRO_PAN significa que el gestor de colas envía una respuesta sólo si el mandato se ejecuta correctamente.
- MQRO_NAN solo significa que el gestor de colas envía una respuesta sólo si el mandato falla.
- Si un mandato se completa con un aviso, se envía una respuesta si se establece MQRO_PAN o MQRO_NAN.
- MQRO_PAN + MQRO_NAN significa que el gestor de colas envía una respuesta tanto si el mandato se ejecuta correctamente como si falla. Esto tiene el mismo efecto desde la perspectiva del gestor de colas que establecer `MsgType` en MQMT_REQUEST.

- Si no se establece MQRO_PAN ni MQRO_NAN, nunca se envía ninguna respuesta.

Format
Establézcalo en MQFMT_RF_HEADER_2

MsgId
Este campo se establece normalmente en MQMI_NONE, de modo que el gestor de colas genera un valor exclusivo.

CorrelId
Este campo se puede establecer en cualquier valor. Si la identidad del remitente incluye un `CorrelId`, especifique este valor, junto con MQRO_PASS_CORREL_ID en el campo `Informe` , para asegurarse de que se establece en todos los mensajes de respuesta enviados por el gestor de colas al remitente.

ReplyToQ
Este campo define la cola a la que se enviarán las respuestas, si las hay. Puede ser la cola del remitente; esto tiene la ventaja de que el parámetro `QName` se puede omitir del mensaje. Sin embargo, si las respuestas se van a enviar a una cola diferente, se necesita el parámetro `QName` .

GestorColasRespuesta
Este campo define el gestor de colas para las respuestas. Si deja este campo en blanco (el valor predeterminado), el gestor de colas local pone su propio nombre en este campo.

Valores de MQMD para publicaciones reenviadas por un gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) cuando envía una publicación a un suscriptor. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Report
`Informe` se establece en MQRO_NONE.

MsgType
`MsgType` se establece en MQMT_DATAGRAM.

Caducidad
`Caducidad` se establece en el valor del mensaje `Publicar` recibido del publicador. En el caso de un mensaje retenido, el tiempo pendiente se reduce en el tiempo aproximado que el mensaje ha estado en el gestor de colas.

Format
`Formato` se establece en MQFMT_RF_HEADER_2

MsgId
`MsgId` se establece en un valor exclusivo.

CorrelId
Si `CorrelId` forma parte de la identidad del suscriptor, este es el valor especificado por el suscriptor al registrarse. De lo contrario, es un valor distinto de cero elegido por el gestor de colas.

Priority
`Prioridad` toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPRI_PRIORITY_AS_Q_DEF.

Persistencia
`Persistencia` toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPER_PERSISTENCE_AS_Q_DEF, a menos que se especifique lo contrario en el mensaje `Registrar suscriptor` para el suscriptor al que se envía esta publicación.

ReplyToQ
`ReplyToQ` se establece en blancos.

GestorColasRespuesta
`ReplyToQMgr` se establece en el nombre del gestor de colas.

UserIdentifier
`UserIdentifier` es el identificador de usuario del suscriptor, tal como se establece cuando se registra el suscriptor.

AccountingToken
`AccountingToken` es la señal de contabilidad del suscriptor, tal como se ha establecido cuando el suscriptor se registró por primera vez.

ApplIdentityData
`Datos deApplIdentity` son los datos de identidad de aplicación del suscriptor, tal como se ha establecido cuando el suscriptor se registró por primera vez.

PutApplType
`PutApplTipo` se establece en MQAT_BROKER.

PutApplName
`PutApplNombre` se establece en los primeros 28 caracteres del nombre del gestor de colas.

PutDate
`PutDate` es la fecha en la que se ha colocado el mensaje.

PutTime
`PutTime` es la hora en que se ha colocado el mensaje.

ApplOriginData
`ApplOriginDatos` se establece en blancos.

Valores de MQMD en mensajes de respuesta del gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) al enviar una respuesta a un mensaje de publicación. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Report
`Informe` se establece en todos los ceros.

MsgType
`MsgType` se establece en MQMT_REPLY.

Format
`Formato` se establece en MQFMT_RF_HEADER_2

MsgId
El valor de `MsgId` depende de las opciones de `Informe` del mensaje de mandato original. De forma predeterminada, se establece en MQMI_NONE, para que el gestor de colas genere un valor exclusivo.

CorrelId
El valor de `CorrelId` depende de las opciones de `Informe` del mensaje de mandato original. De forma predeterminada, esto significa que el `CorrelId` se establece en el mismo valor que el `MsgId` del mensaje de mandato. Esto se puede utilizar para correlacionar mandatos con sus respuestas.

Priority
`Prioridad` se establece en el mismo valor que en el mensaje de mandato original.

Persistencia
`Persistence` se establece en el valor establecido en el mensaje de mandato original.

Caducidad
`Caducidad` se establece en el mismo valor que en el mensaje de mandato original recibido por el gestor de colas.

PutApplType
`PutApplTipo` se establece en MQAT_BROKER.

PutApplName
`PutApplNombre` se establece en los primeros 28 caracteres del nombre del gestor de colas.

Otros campos de contexto se establecen como si se hubieran generado con MQPMO_PASS_IDENTITY_CONTEXT.

Codificaciones de máquina

Esta sección describe la estructura del campo *Encoding* en el descriptor de mensaje.

Consulte “MQMD - Descriptor de mensaje” en la página 421 para obtener un resumen de los campos de la estructura.

El campo *Encoding* es un entero de 32 bits que se divide en cuatro subcampos separados; estos subcampos identifican:

- La codificación utilizada para enteros binarios
- La codificación utilizada para enteros de decimal empaquetado
- La codificación utilizada para números de coma flotante
- Bits reservados

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Se definen las máscaras siguientes:

MÁSCARA_ENTERO_MQENC_MÁSCARA
Máscara para codificación de enteros binarios.

Este subcampo ocupa las posiciones de bits 28 a 31 dentro del campo *Encoding* .

MÁSCARA_DECIMAL_MQENC_MÁSCARA
Máscara para codificación de enteros decimales empaquetados.

Este subcampo ocupa las posiciones de bits 24 a 27 dentro del campo *Encoding* .

Máscara MQENC_FLOAT_MASK
Máscara para codificación de coma flotante.

Este subcampo ocupa las posiciones de bits 20 a 23 dentro del campo *Encoding* .

MÁSCARA_RESERVA_MQENC_RESERVADO
Máscara para bits reservados.

Este subcampo ocupa las posiciones de bits 0 a 19 dentro del campo *Encoding* .

Codificación de enteros binarios

Los valores siguientes son válidos para la codificación de entero binario:

MQENC_INTEGER_UNDEFINED
Los enteros binarios se representan utilizando una codificación que no está definida.

MQENC_INTEGER_NORMAL
Los enteros binarios se representan de la forma convencional:

- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

MQENC_INTEGER_REVERSED
Los enteros binarios se representan de la misma forma que MQENC_INTEGER_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_INTEGER_NORMAL.

Codificación de enteros decimales empaquetados

Los valores siguientes son válidos para la codificación de enteros decimales empaquetados:

MQENC_DECIMAL_UNDEFINED
Los enteros de decimal empaquetado se representan utilizando una codificación que no está definida.

MQENC_DECIMAL_NORMAL
Los enteros decimales empaquetados se representan de la forma convencional:

- Cada dígito decimal en la forma imprimible del número se representa en decimal empaquetado por un dígito hexadecimal único en el rango de X' 0 'a X' 9'. Cada dígito hexadecimal ocupa cuatro bits, por lo que cada byte del número decimal empaquetado representa dos dígitos decimales en la forma imprimible del número.
- El byte menos significativo del número decimal empaquetado es el byte que contiene el dígito decimal menos significativo. Dentro de ese byte, los cuatro bits más significativos contienen el dígito decimal menos significativo, y los cuatro bits menos significativos contienen el signo. El signo es X'C '(positivo), X 'D' (negativo) o X'F ' (sin signo).
- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior.

MQENC_DECIMAL_REVERSED
Los enteros decimales empaquetados se representan de la misma forma que MQENC_DECIMAL_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_DECIMAL_NORMAL.

Codificación de coma flotante

Los valores siguientes son válidos para la codificación de coma flotante:

MQENC_FLOAT_UNDEFINED
Los números de coma flotante se representan utilizando una codificación que no está definida.

MQENC_FLOAT_IEEE_NORMAL
Los números de coma flotante se representan utilizando el IEEE estándar[4]formato de coma flotante, con los bytes ordenados de la siguiente manera:

- El byte menos significativo de la mantisa tiene la dirección más alta de cualquiera de los bytes del número; el byte que contiene el exponente tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

Los detalles de la codificación flotante IEEE se pueden encontrar en el estándar IEEE 754.

MQENC_FLOAT_IEEE_REVERSED
Los números de coma flotante se representan de la misma forma que MQENC_FLOAT_IEEE_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390
Los números de coma flotante se representan utilizando el formato de coma flotante estándar System/390 ; también lo utiliza System/370.

Construcción de codificaciones

Para construir un valor para el campo *Encoding* en MQMD, las constantes relevantes que describen las codificaciones necesarias se pueden añadir juntas (no añadir la misma constante más de una vez) o combinarse utilizando la operación OR a nivel de bit (si el lenguaje de programación soporta operaciones de bits).

Sea cual sea el método utilizado, combine sólo una de las codificaciones MQENC_INTEGER_ * con una de las codificaciones MQENC_DECIMAL_ * y una de las codificaciones MQENC_FLOAT_ *.

[4] El Instituto de Ingenieros Eléctricos y Electrónicos

Análisis de codificaciones

El campo *Encoding* contiene subcampos; debido a esto, las aplicaciones que necesitan examinar la codificación entera, decimal empaquetado o flotante deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
   - MQENC_INTEGER_MASK para la codificación de enteros binarios
   - MQENC_DECIMAL_MASK para la codificación de enteros empaquetados
   - MQENC_FLOAT_MASK para la codificación de coma flotante

   Llame al valor A.
2. Combine el campo *Encoding* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. B es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
   - 1 para la codificación de enteros binarios
   - 16 para la codificación de enteros decimales empaquetados
   - 256 para la codificación de coma flotante

   Llame al valor A.
2. Divida el valor del campo *Encoding* por A ; llamar al resultado B.
3. Divida B por 16; llame al resultado C.
4. Multiplique C por 16 y reste de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. E es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Resumen de codificaciones de arquitectura de máquina

Las codificaciones para arquitecturas de máquina se muestran en Tabla 631 en la página 918.

Tabla 631. Resumen de codificaciones para arquitecturas de máquina

Arquitectura de máquina	Codificación de enteros binarios	Codificación de entero decimal empaquetado	Codificación de coma flotante
IBM i	normal	normal	IEEE normal
Intel x86	reversed	reversed	IEEE invertido
PowerPC	normal	normal	IEEE normal
System/390	normal	normal	System/390

Opciones de informe y distintivos de mensaje

En esta sección se describen los campos *Report* y *MsgFlags* que forman parte del MQMD del descriptor de mensaje especificado en las llamadas MQGET, MQPUT y MQPUT1 .

Los temas de esta sección describen:

- La estructura del campo de informe y cómo lo procesa el gestor de colas
- Cómo analiza una aplicación el campo de informe
- La estructura del campo de distintivos de mensaje

Para obtener más información sobre el descriptor de mensaje MQMD, consulte "MQMD - Descriptor de mensaje" en la página 421.

Estructura del campo de informe

Esta información describe la estructura del campo de informe.

El campo *Report* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Opciones de informe que se rechazan si el gestor de colas local no las reconoce
- Opciones de informe que siempre se aceptan, incluso si el gestor de colas local no las reconoce
- Opciones de informe que sólo se aceptan si se cumplen otras condiciones

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits de un subcampo no son necesariamente adyacentes. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQRO_REJECT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe no soportadas por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_REPORT_OPTIONS_ERROR.

Este subcampo ocupa las posiciones de bits 3 y 11 a 13.

MQRO_ACCEPT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 . En este caso, se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION.

Este subcampo ocupa las posiciones de bits 0 a 2, 4 a 10 y 24 a 31.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_ACTIVITY
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- MQRO_NEW_MSG_ID

- MQRO_NONE
- MQRO_PAN
- MQRO_PASS_CORREL_ID
- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK

Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMgrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_REPORT_OPTIONS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 14 a 23.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Si se especifica alguna opción en el campo *Report* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *Report* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Si se devuelve MQCC_WARNING, no se define qué código de razón se devuelve si existen otras condiciones de aviso.

La posibilidad de especificar y tener opciones de informe aceptadas que no son reconocidas por el gestor de colas local es útil cuando se envía un mensaje con una opción de informe que es reconocida y procesada por un gestor de colas *remoto* .

Análisis del campo de informe

El campo *Report* contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:
   - Informe MQRO_COA_WITH_FULL_DATA para COA
   - Informe MQRO_COD_WITH_FULL_DATA para COD
   - MQRO_EXCEPTION_WITH_FULL_DATA para informe de excepciones
   - MQRO_EXPIRATION_WITH_FULL_DATA para informe de caducidad

   Llame al valor A.

En z/OS, utilice los valores MQRO_ * _WITH_DATA en lugar de los valores MQRO_ * _WITH_FULL_DATA.

2. Combine el campo *Report* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. Pruebe B para la igualdad con cada valor que sea posible para ese tipo de informe.

   Por ejemplo, si A es MQRO_EXCEPTION_WITH_FULL_DATA, pruebe la igualdad de B con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como el valor A cualquiera de estas dos constantes que sea apropiado y, a continuación, continúe como se ha descrito anteriormente.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:

   - Informe MQRO_COA para COA
   - Informe MQRO_COD para COD
   - MQRO_EXCEPTION para el informe de excepciones
   - MQRO_EXPIRATION para informe de caducidad

   Llame al valor A.
2. Divida el campo *Report* por A ; llamar al resultado B.
3. Divida B por 8 ; llamar al resultado C.
4. Multiplique C por 8 y resta de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. Pruebe E para la igualdad con cada valor que sea posible para ese tipo de informe.

   Por ejemplo, si A es MQRO_EXCEPTION, pruebe la igualdad de E con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

El pseudocódigo siguiente ilustra esta técnica para los mensajes de informe de excepción:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como el valor A cualquiera de estas dos constantes es apropiado y, a continuación, continúe como se ha descrito anteriormente, pero sustituyendo el valor 8 en los pasos anteriores por el valor 2.

Estructura del campo de distintivos de mensaje

Esta información describe la estructura del campo de distintivos de mensaje.

El campo *MsgFlags* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Distintivos de mensaje que se rechazan si el gestor de colas local no los reconoce
- Distintivos de mensajes que siempre se aceptan, incluso si el gestor de colas local no los reconoce
- Distintivos de mensaje que se aceptan sólo si se cumplen otras condiciones

Nota: Todos los subcampos de *MsgFlags* están reservados para que los utilice el gestor de colas.

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQMF_REJECT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_MSG_FLAGS_ERROR.

Este subcampo ocupa las posiciones de bits 20 a 31.

En este subcampo se incluyen los distintivos de mensaje siguientes:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBIDO

MQMF_ACCEPT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 . El código de terminación es MQCC_OK.

Este subcampo ocupa las posiciones de bits 0 a 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMgrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_OK si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_MSG_FLAGS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 12 a 19.

Si hay distintivos especificados en el campo *MsgFlags* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MsgFlags* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

salida de conversión de datos

Esta colección de temas describe la interfaz para la salida de conversión de datos y el proceso realizado por el gestor de colas cuando es necesaria la conversión de datos.

Para obtener más información sobre la conversión de datos, consulte *Conversión de datos en IBM MQ* en https://www.ibm.com/support/pages/node/317869.

La salida de conversión de datos se invoca como parte del proceso de la llamada MQGET para convertir los datos del mensaje de aplicación a la representación que necesita la aplicación receptora. La conversión de los datos del mensaje de aplicación es opcional; requiere que se especifique la opción MQGMO_CONVERT en la llamada MQGET.

Se describen los siguientes temas:

- El proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT; consulte "Proceso de conversión" en la página 923.
- Convenios de proceso utilizados por el gestor de colas al procesar un formato incorporado; estos convenios también se recomiendan para las salidas escritas por el usuario. Consulte "Convenios de proceso" en la página 924.
- Consideraciones especiales para convertir mensajes de informe; consulte "Conversión de mensajes de informe" en la página 929.
- Los parámetros pasados a la salida de conversión de datos; consulte "MQ_DATA_CONV_EXIT-Salida de conversión de datos" en la página 942.
- Una llamada que se puede utilizar desde la salida para convertir datos de caracteres entre distintas representaciones; consulte "MQXCNVC-Convertir caracteres" en la página 935.
- El parámetro de estructura de datos que es específico de la salida; consulte "MQDXP-Parámetro de salida de conversión de datos" en la página 929.

Proceso de conversión

Esta información describe el proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT.

El gestor de colas realiza las acciones siguientes si se especifica la opción MQGMO_CONVERT en la llamada MQGET y hay un mensaje que se debe devolver a la aplicación:

1. Si se cumple una o más de las condiciones siguientes, no es necesaria ninguna conversión:
   - Los datos del mensaje ya están en el juego de caracteres y la codificación que necesita la aplicación que emite la llamada MQGET. La aplicación debe establecer los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** de la llamada MQGET en los valores necesarios, antes de emitir la llamada.
   - La longitud de los datos del mensaje es cero.
   - La longitud del parámetro **Buffer** de la llamada MQGET es cero.

   En estos casos, el mensaje se devuelve sin conversión a la aplicación que emite la llamada MQGET; los valores *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** se establecen en los valores de la información de control del mensaje y la llamada se completa con una de las siguientes combinaciones de código de terminación y código de razón:

Tabla 632. Combinaciones de código de terminación y código de razón

Código de terminación	Código de razón
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_WARNING	MQRC_TRUNCATED_MSG_FAILED

Los pasos siguientes sólo se realizan si el juego de caracteres o la codificación de los datos del mensaje difiere del valor correspondiente en el parámetro **MsgDesc** y hay datos que se deben convertir:

2. Si el campo *Format* de la información de control del mensaje tiene el valor MQFMT_NONE, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

   En todos los demás casos, el proceso de conversión continúa.

3. El mensaje se elimina de la cola y se coloca en un almacenamiento intermedio temporal que tiene el mismo tamaño que el parámetro **Buffer** . Para las operaciones de examen, el mensaje se copia en el almacenamiento intermedio temporal, en lugar de eliminarse de la cola.
4. Si el mensaje debe truncarse para que quepa en el almacenamiento intermedio, se realiza lo siguiente:
   - Si no se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED.
   - Si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG , el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.
5. Si el mensaje se puede acomodar en el almacenamiento intermedio sin truncamiento, o se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, se hace lo siguiente:
   - Si el formato es un formato incorporado, el almacenamiento intermedio se pasa al servicio de conversión de datos del gestor de colas.
   - Si el formato no es un formato incorporado, el almacenamiento intermedio se pasa a una salida escrita por el usuario con el mismo nombre que el formato. Si no se puede encontrar la salida, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

   Si no se produce ningún error, la salida del servicio de conversión de datos o de la salida escrita por el usuario es el mensaje convertido, más el código de terminación y el código de razón que se devolverá a la aplicación que emite la llamada MQGET.

6. Si la conversión es satisfactoria, el gestor de colas devuelve el mensaje convertido a la aplicación. En este caso, el código de terminación y el código de razón devueltos por la llamada MQGET son una de las combinaciones siguientes:

Tabla 633. Combinaciones de código de terminación y código de razón

Código de terminación	Código de razón
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED

Sin embargo, si la conversión la realiza una salida escrita por el usuario, se pueden devolver otros códigos de razón, incluso cuando la conversión es satisfactoria.

Si la conversión falla, el gestor de colas devuelve el mensaje sin convertir a la aplicación, con los campos *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** establecidos en los valores de la información de control del mensaje y con el código de terminación MQCC_WARNING.

Convenios de proceso

Al convertir un formato incorporado, el gestor de colas sigue los convenios de proceso descritos.

Las salidas escritas por el usuario también deben seguir estas convenciones, aunque el gestor de colas no las aplica. Los formatos incorporados convertidos por el gestor de colas son:

- MQFMT_ADMIN
- MQFMT_CICS (solo z/OS)

- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT versión 1
- MQFMT_EVENT versión 2
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER (solo z/OS)
- MQFMT_XMIT_Q_HEADER

1. Si el mensaje se expande durante la conversión y supera el tamaño del parámetro **Buffer** , se realiza lo siguiente:
   - Si no se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG.
   - Si se especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG *era* , el mensaje se trunca, el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.
2. Si se produce un truncamiento (antes o durante la conversión), el número de bytes válidos devueltos en el parámetro **Buffer** puede ser menor que la longitud del almacenamiento intermedio.

   Esto puede ocurrir, por ejemplo, si un entero de 4 bytes o un carácter DBCS se encuentra entre los extremos del almacenamiento intermedio. El elemento de información incompleto no se convierte y los bytes del mensaje devuelto no contienen información válida. Esto también puede ocurrir si un mensaje que se ha truncado antes de la conversión se reduce durante la conversión.

   Si el número de bytes válidos devueltos es menor que la longitud del almacenamiento intermedio, los bytes no utilizados al final del almacenamiento intermedio se establecen en nulos.
3. Si una matriz o serie se encuentra entre el final del almacenamiento intermedio, se convierten tantos datos como sea posible; sólo no se convierte el elemento de matriz concreto o el carácter DBCS que está incompleto; se convierten los elementos o caracteres de matriz anteriores.
4. Si se produce un truncamiento (antes o durante la conversión), la longitud devuelta para el parámetro **DataLength** es la longitud del mensaje sin convertir antes del truncamiento.
5. Cuando las series se convierten entre juegos de caracteres de un solo byte (SBCS), juegos de caracteres de doble byte (DBCS) o juegos de caracteres de varios bytes (MBCS), las series pueden expandirse o contraerse.
   - En los formatos PCF MQFMT_ADMIN, MQFMT_EVENT y MQFMT_PCF, las series de las estructuras MQCFST y MQCFSL se expanden o contraen según sea necesario para acomodar la serie después de la conversión.

     Para la estructura de lista de series MQCFSL, las series de la lista pueden expandirse o contraerse en cantidades diferentes. Si esto sucede, el gestor de colas rellena las series más cortas con espacios en blanco para que tengan la misma longitud que la serie más larga después de la conversión.

- En el formato MQFMT_REF_MSG_HEADER, las series a las que se dirigen los campos `SrcEnvOffset`, `SrcNameOffset`, `DestEnvOffset`y `DestNameOffset` se expanden o contraen según sea necesario para acomodar las series después de la conversión.
- En el formato MQFMT_RF_HEADER, el campo `NameValueString` se expande o contrae según sea necesario para acomodar los pares nombre-valor después de la conversión.
- En estructuras con tamaños de campo fijos, el gestor de colas permite que las series se expandan o se contraigan dentro de sus campos fijos, siempre que no se pierda información significativa. En este sentido, los espacios en blanco finales y los caracteres que siguen al primer carácter nulo del campo se tratan como insignificantes.
  - Si la serie se expande, pero sólo es necesario descartar caracteres insignificantes para acomodar la serie convertida en el campo, la conversión se realiza correctamente y la llamada se completa con MQCC_OK y el código de razón MQRC_NONE (suponiendo que no haya otros errores).
  - Si la serie se expande, pero la serie convertida requiere que se descarten caracteres significativos para que quepan en el campo, el mensaje se devuelve sin convertir y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_STRING_TOO_BIG.

    Nota: El código de razón MQRC_CONVERTED_STRING_TOO_BIG da como resultado en este caso si se ha especificado o no la opción MQGMO_ACCEPT_TRUNCATED_MSG.
  - Si la serie se contrae, el gestor de colas rellena la serie con espacios en blanco hasta la longitud del campo.

6. Para los mensajes que constan de una o más estructuras de cabecera MQ seguidas de datos de usuario, es posible que se conviertan una o más de las estructuras de cabecera, mientras que el resto del mensaje no lo es. Sin embargo, (con dos excepciones) los campos *`CodedCharSetId`* y *`Encoding`* de cada estructura de cabecera siempre indican correctamente el juego de caracteres y la codificación de los datos que siguen a la estructura de cabecera.

   Las dos excepciones son las estructuras MQCIH y MQIIH, donde los valores de los campos *`CodedCharSetId`* y *`Encoding`* de esas estructuras no son significativos. Para esas estructuras, los datos que siguen a la estructura están en el mismo juego de caracteres y codificación que la propia estructura MQCIH o MQIIH.

7. Si los campos *`CodedCharSetId`* o *`Encoding`* de la información de control del mensaje que se está recuperando, o en el parámetro **MsgDesc** , especifican valores que no están definidos o no están soportados, el gestor de colas puede ignorar el error si no es necesario utilizar el valor no definido o no soportado al convertir el mensaje.

   Por ejemplo, si el campo *`Encoding`* del mensaje especifica una codificación flotante no soportada, pero el mensaje contiene sólo datos enteros, o contiene datos de coma flotante que no requieren conversión (porque las codificaciones flotante de origen y destino son idénticas), es posible que el error no se diagnostique.

   Si se diagnostica el error, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y uno de los códigos de razón MQRC_SOURCE_ * _ERROR o MQRC_TARGET_ * _ERROR (según corresponda); los campos *`CodedCharSetId`* y *`Encoding`* del parámetro **MsgDesc** se establecen en los valores de la información de control del mensaje.

   Si el error no se diagnostica y la conversión se completa correctamente, los valores devueltos en los campos *`CodedCharSetId`* y *`Encoding`* en el parámetro **MsgDesc** son los especificados por la aplicación que emite la llamada MQGET.

8. En todos los casos, si el mensaje se devuelve a la aplicación sin convertir, el código de finalización se establece en MQCC_WARNING, y los campos *`CodedCharSetId`* y *`Encoding`* del parámetro **MsgDesc** se establecen en los valores adecuados para los datos sin convertir. Esto también se realiza para MQFMT_NONE.

   El parámetro **Reason** se establece en un código que indica por qué no se ha podido llevar a cabo la conversión, a menos que el mensaje también se haya tenido que truncar; los códigos de razón relacionados con el truncamiento tienen prioridad sobre los códigos de razón relacionados con la conversión. (Para determinar si se ha convertido un mensaje truncado, compruebe los valores devueltos en los campos *`CodedCharSetId`* y *`Encoding`* en el parámetro **MsgDesc** .)

Cuando se diagnostica un error, se devuelve un código de razón específico o el código de razón general MQRC_NOT_CONVERT. El código de razón devuelto depende de las prestaciones de diagnóstico del servicio de conversión de datos subyacente.

9. Si se devuelve el código de terminación MQCC_WARNING y hay más de un código de razón relevante, el orden de prioridad es el siguiente:
   a. Las razones siguientes tienen prioridad sobre todas las demás; sólo puede surgir una de las razones de este grupo:
      - MQRC_SIGNAL_REQUEST_ACCEPTED
      - MQRC_TRUNCATED_MSG_ACCEPTED
   b. El orden de prioridad dentro de los códigos de razón restantes no está definido.
10. Al finalizar la llamada MQGET:
    - El siguiente código de razón indica que el mensaje se ha convertido correctamente:
      – MQRC_NONE
    - Los siguientes códigos de razón indican que el mensaje *podría* haberse convertido correctamente (compruebe los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** para averiguarlo):
      – MQRC_MSG_MARKED_BROWSE_CO_OP
      – MQRC_TRUNCATED_MSG_ACCEPTED
    - Los demás códigos de razón indican que el mensaje no se ha convertido.

    El siguiente proceso es específico de los formatos incorporados; no se aplica a los formatos definidos por el usuario:
11. Con la excepción de los formatos siguientes:
    - MQFMT_ADMIN
    - MQFMT_COMMAND_1
    - MQFMT_COMMAND_2
    - MQFMT_EVENT
    - MQFMT_IMS_VAR_STRING
    - MQFMT_PCF
    - MQFMT_STRING

    ninguno de los formatos incorporados se puede convertir de o a conjuntos de caracteres que no tengan caracteres SBCS para los caracteres que son válidos en los nombres de cola. Si se intenta realizar una conversión de este tipo, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR, según corresponda.

    El juego de caracteres Unicode UTF-16 es un ejemplo de un juego de caracteres que no tiene caracteres SBCS para los caracteres que son válidos en los nombres de cola.
12. Si los datos de mensaje para un formato incorporado se truncan, los campos dentro del mensaje que contienen longitudes de series, o recuentos de elementos o estructuras, no se ajustan para reflejar la longitud de los datos devueltos realmente a la aplicación; los valores devueltos para dichos campos dentro de los datos del mensaje son los valores aplicables al mensaje *antes del truncamiento*.

    Al procesar mensajes como, por ejemplo, un mensaje MQFMT_ADMIN truncado, asegúrese de que la aplicación no intenta acceder a los datos más allá del final de los datos devueltos.
13. Si el nombre de formato es MQFMT_DEAD_LETTER_HEADER, los datos de mensaje empiezan con una estructura MQDLH, posiblemente seguida de cero o más bytes de datos de mensaje de aplicación. El formato, el juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos `Format`, `CodedCharSetId`y `Encoding` en la estructura MQDLH al principio del mensaje. Debido a que la estructura MQDLH y los datos de mensajes de aplicación

pueden tener diferentes conjuntos de caracteres y codificaciones, uno, otro o ambos de la estructura MQDLH y los datos de mensajes de aplicación pueden requerir conversión.

El gestor de colas convierte primero la estructura MQDLH, según sea necesario. Si la conversión es satisfactoria, o la estructura MQDLH no requiere conversión, el gestor de colas comprueba los campos `CodedCharSetId` y `Encoding` de la estructura MQDLH para ver si es necesaria la conversión de los datos del mensaje de aplicación. Si la conversión es necesaria, el gestor de colas invoca la salida escrita por el usuario con el nombre proporcionado por el campo `Format` en la estructura MQDLH, o realiza la propia conversión (si `Format` es el nombre de un formato incorporado).

Si la llamada MQGET devuelve un código de terminación de MQCC_WARNING, y el código de razón es uno de los que indica que la conversión no ha sido satisfactoria, se aplica uno de los siguientes:

- No se ha podido convertir la estructura MQDLH. En este caso, los datos del mensaje de aplicación tampoco se habrán convertido.
- La estructura MQDLH se ha convertido, pero los datos del mensaje de aplicación no.

La aplicación puede examinar los valores devueltos en los campos `CodedCharSetId` y `Encoding` en el parámetro **MsgDesc** , y los de la estructura MQDLH, para determinar cuál de los anteriores se aplica.

14. Si el nombre de formato es MQFMT_XMIT_Q_HEADER, los datos del mensaje empiezan con una estructura MQXQH, posiblemente seguida de cero o más bytes de datos adicionales. Estos datos adicionales suelen ser los datos de mensaje de aplicación (que pueden ser de longitud cero), pero también puede haber una o más estructuras de cabecera de MQ adicionales presentes, al principio de los datos adicionales.

    La estructura MQXQH debe estar en el juego de caracteres y la codificación del gestor de colas. El formato, el juego de caracteres y la codificación de los datos que siguen a la estructura MQXQH se proporcionan mediante los campos `Format`, `CodedCharSetId`y `Encoding` en la estructura MQMD contenida en MQXQH. Para cada estructura de cabecera MQ posterior presente, los campos `Format`, `CodedCharSetId`y `Encoding` de la estructura describen los datos que siguen a esa estructura; esos datos son otra estructura de cabecera MQ o los datos de mensaje de aplicación.

    Si se especifica la opción MQGMO_CONVERT para un mensaje MQFMT_XMIT_Q_HEADER, los datos del mensaje de aplicación y algunas de las estructuras de cabecera de MQ se convierten, *pero los datos de la estructura MQXQH no son*. Al volver de la llamada MQGET, por lo tanto:

    - Los valores de los campos `Format`, `CodedCharSetId`y `Encoding` del parámetro **MsgDesc** describen los datos de la estructura MQXQH y no los datos del mensaje de aplicación; por lo tanto, los valores no son los mismos que los especificados por la aplicación que ha emitido la llamada MQGET.

      El efecto de esto es que una aplicación que obtiene repetidamente mensajes de una cola de transmisión con la opción MQGMO_CONVERT especificada debe restablecer los campos `CodedCharSetId` y `Encoding` en el parámetro **MsgDesc** a los valores necesarios para los datos de mensaje de aplicación, antes de cada llamada MQGET.
    - Los valores de los campos `Format`, `CodedCharSetId`y `Encoding` de la última estructura de cabecera de MQ presente describen los datos del mensaje de aplicación. Si no hay otras estructuras de cabecera MQ presentes, los datos del mensaje de aplicación se describen mediante estos campos en la estructura MQMD dentro de la estructura MQXQH. Si la conversión es satisfactoria, los valores serán los mismos que los especificados en el parámetro **MsgDesc** por la aplicación que ha emitido la llamada MQGET.

    Si el mensaje es un mensaje de lista de distribución, la estructura MQXQH va seguida de una estructura MQDH (más sus matrices de registros MQOR y MQPMR), que a su vez puede ir seguida de cero o más estructuras de cabecera MQ adicionales y de cero o más bytes de datos de mensaje de aplicación. Al igual que la estructura MQXQH, la estructura MQDH debe estar en el juego de caracteres y la codificación del gestor de colas, y no se convierte en la llamada MQGET, incluso si se especifica la opción MQGMO_CONVERT.

El proceso de las estructuras MQXQH y MQDH descritas anteriormente está pensado principalmente para que las utilicen los agentes de canal de mensajes cuando obtienen mensajes de las colas de transmisión.

Conversión de mensajes de informe

En general, un mensaje de informe puede contener cantidades variables de datos de mensaje de aplicación, según las opciones de informe especificadas por el remitente del mensaje original. Sin embargo, un informe de actividad puede contener datos pero sin que la opción de informe mencione * _WITH_DATA en la constante.

En concreto, un mensaje de informe puede contener:

1. No hay datos de mensaje de aplicación
2. Algunos de los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_DATA y el mensaje tiene más de 100 bytes.
3. Todos los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_FULL_DATA, o especifica MQRO_ * _WITH_DATA y el mensaje tiene 100 bytes o menos.

Cuando el gestor de colas o el agente de canal de mensajes genera un mensaje de informe, copia el nombre de formato del mensaje original en el campo *Format* de la información de control del mensaje de informe. Por lo tanto, el nombre de formato del mensaje de informe puede implicar una longitud de datos diferente de la longitud realmente presente en el mensaje de informe (casos 1 y 2 anteriores).

Si se especifica la opción MQGMO_CONVERT cuando se recupera el mensaje de informe:

- Para el caso 1 anterior, no se invoca la salida de conversión de datos (porque el mensaje de informe no tiene datos).
- Para el caso 3 anterior, el nombre de formato implica correctamente la longitud de los datos del mensaje.
- Pero para el caso 2 anterior, se invoca la salida de conversión de datos para convertir un mensaje que es *más corto* que la longitud implícita por el nombre de formato.

  Además, el código de razón pasado a la salida suele ser MQRC_NONE (es decir, el código de razón no indica que el mensaje se haya truncado). Esto sucede porque los datos del mensaje han sido truncados por el *emisor* del mensaje de informe y no por el gestor de colas del receptor en respuesta a la llamada MQGET.

Debido a estas posibilidades, la salida de conversión de datos no debe utilizar el nombre de formato para deducir la longitud de los datos que se le pasan; en su lugar, la salida debe comprobar la longitud de los datos proporcionados y estar preparada para convertir menos datos que la longitud implícita por el nombre de formato. Si los datos se pueden convertir correctamente, la salida debe devolver el código de terminación MQCC_OK y el código de razón MQRC_NONE. La longitud de los datos de mensaje que se van a convertir se pasa a la salida como parámetro **InBufferLength** .

Interfaz de programación sensible al producto

MQDXP-Parámetro de salida de conversión de datos

La estructura MQDXP es un parámetro que el gestor de colas pasa a la salida de conversión de datos cuando se invoca la salida para convertir los datos del mensaje como parte del proceso de la llamada MQGET. Consulte la descripción de la llamada MQ_DATA_CONV_EXIT para obtener detalles de la salida de conversión de datos.

Los datos de tipo carácter en MQDXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId** . Los datos numéricos en MQDXP están en la codificación de máquina nativa; esto lo proporciona MQENC_NATIVE.

La salida sólo puede cambiar los campos *DataLength*, *CompCode*, *Reason*y *ExitResponse* en MQDXP; los cambios en otros campos se ignoran. Sin embargo, el campo *DataLength* no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

Cuando el control vuelve al gestor de colas desde la salida, el gestor de colas comprueba los valores devueltos en MQDXP. Si los valores devueltos no son válidos, el gestor de colas continúa el proceso como si la salida hubiera devuelto MQXDR_CONVERSION_FAILED en *ExitResponse* ; sin embargo, el gestor de colas ignora los valores de los campos *CompCode* y *Reason* devueltos por la salida en este caso, y utiliza en su lugar los valores que esos campos tenían en *entrada* para la salida. Los siguientes valores en MQDXP hacen que se produzca este proceso:

- El campo *ExitResponse* no es MQXDR_OK y no es MQXDR_CONVERSION_FAILED
- El campo *CompCode* no es MQCC_OK y no es MQCC_WARNING
- *DataLength* campo menor que cero, o *DataLength* campo cambiado cuando el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

La tabla siguiente resume los campos de la estructura.

Tabla 634. Campos en MQDXP

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
AppOptions	Opciones de aplicación	AppOptions
Encoding	Codificación numérica necesaria para la aplicación	Codificación
CodedCharSetId	Juego de caracteres necesario para la aplicación	CodedCharSetId
DataLength	Longitud en bytes de datos de mensaje	DataLength
CompCode	Código de terminación	CompCode
Reason	Código de razón que califica *CompCode*	Razón
ExitResponse	Respuesta de la salida	ExitResponse
Hconn	Descriptor de contexto de conexión	Hconn
pEntryPoints	Dirección de la estructura MQIEP	PuntospEntry

Campos

La estructura MQDXP contiene los campos siguientes; los campos se describen en orden alfabético.

AppOptions

Tipo: MQLONG

Se trata de una copia del campo *Options* de la estructura MQGMO especificada por la aplicación que emite la llamada MQGET. Es posible que la salida tenga que examinarlos para determinar si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG.

Es un campo de entrada para la salida.

CodedCharSetId

Tipo: MQLONG

Este es el identificador de juego de caracteres codificado del juego de caracteres que necesita la aplicación que emite la llamada MQGET; consulte el campo *CodedCharSetId* en la estructura MQMD para obtener más detalles. Si la aplicación especifica el valor especial MQCCSI_Q_MGR en la llamada MQGET, el gestor de colas lo cambia por el identificador de juego de caracteres real del juego de caracteres utilizado por el gestor de colas, antes de invocar la salida.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *CodedCharSetId* del descriptor de mensaje.

Es un campo de entrada para la salida.

CompCode

Tipo: MQLONG

Cuando se invoca la salida, contiene el código de terminación que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida no hace nada. Siempre es MQCC_WARNING, porque el mensaje se ha truncado o el mensaje requiere conversión y esto todavía no se ha realizado.

En la salida de la salida, este campo contiene el código de terminación que se devolverá a la aplicación en el parámetro **CompCode** de la llamada MQGET; sólo son válidos MQCC_OK y MQCC_WARNING. Consulte la descripción del campo *Reason* para obtener sugerencias sobre cómo la salida puede establecer este campo en la salida.

Es un campo de entrada/salida para la salida.

DataLength

Tipo: MQLONG

Cuando se invoca la salida, este campo contiene la longitud original de los datos del mensaje de aplicación. Si el mensaje se ha truncado para que quepa en el almacenamiento intermedio proporcionado por la aplicación, el tamaño del mensaje proporcionado a la salida es *menor* que el valor de *DataLength*. El tamaño del mensaje proporcionado a la salida siempre lo proporciona el parámetro **InBufferLength** de la salida, independientemente de cualquier truncamiento que se haya producido.

El truncamiento se indica mediante el campo *Reason* que tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada de la salida.

La mayoría de las conversiones no necesitan cambiar esta longitud, pero una salida puede hacerlo si es necesario; el valor establecido por la salida se devuelve a la aplicación en el parámetro **DataLength** de la llamada MQGET. Sin embargo, esta longitud no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico. Esto se debe a que cambiar la longitud haría que los desplazamientos de segmentos posteriores en el mensaje lógico fueran incorrectos.

Tenga en cuenta que, si la salida desea cambiar la longitud de los datos, tenga en cuenta que el gestor de colas ya ha decidido si los datos del mensaje se ajustan al almacenamiento intermedio de la aplicación, basándose en la longitud de los datos *sin convertir* . Esta decisión determina si el mensaje se elimina de la cola (o se mueve el cursor para examinar, para una solicitud de examinar) y no se ve afectado por ningún cambio en la longitud de los datos causado por la conversión. Por este motivo, se recomienda que las salidas de conversión no provoquen un cambio en la longitud de los datos del mensaje de aplicación.

Si la conversión de caracteres implica un cambio de longitud, una serie se puede convertir en otra serie con la misma longitud en bytes, truncando los espacios en blanco finales o rellenando con espacios en blanco según sea necesario.

La salida no se invoca si el mensaje no contiene datos de mensaje de aplicación; por lo tanto, *DataLength* siempre es mayor que cero.

Es un campo de entrada/salida para la salida.

Encoding
Tipo: MQLONG

Codificación numérica necesaria para la aplicación.

Esta es la codificación numérica que necesita la aplicación que emite la llamada MQGET; consulte el campo *Encoding* en la estructura MQMD para obtener más detalles.

Si la conversión es satisfactoria, la salida la copia en el campo *Encoding* del descriptor de mensaje.

Es un campo de entrada para la salida.

ExitOptions
Tipo: MQLONG

Se trata de un campo reservado; su valor es `0`.

ExitResponse
Tipo: MQLONG

Respuesta de la salida.Esto lo establece la salida para indicar el éxito o no de la conversión. Debe ser uno de los siguientes:

MQXDR_Correcto
La conversión ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en la salida de la salida
- El contenido del almacenamiento intermedio de salida de la salida *OutBuffer*. El número de bytes devueltos es el menor del parámetro **OutBufferLength** de la salida y el valor del campo *DataLength* en la salida de la salida.

  Si los campos *Encoding* y *CodedCharSetId* del parámetro de descriptor de mensaje de la salida *ambos* no se modifican, el gestor de colas devuelve:
- El valor de los campos *Encoding* y *CodedCharSetId* en la estructura MQDXP en *entrada* a la salida.

  Si se ha cambiado uno o ambos campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida, el gestor de colas devuelve:
- El valor de los campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida en la salida de la salida

MQXDR_CONVERSION_FAILED
La conversión no ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en *entrada* a la salida
- El contenido del almacenamiento intermedio de entrada de la salida *InBuffer*. El número de bytes devueltos lo proporciona el parámetro **InBufferLength**

Si la salida ha modificado *InBuffer*, los resultados no están definidos.

ExitResponse es un campo de salida de la salida.

Hconn
Tipo: MQHCONN

Se trata de un descriptor de conexión que se puede utilizar en la llamada MQXCNVC. Este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

pEntryPoints
Tipo: PMQIEP

La dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas MQI y DCI.

Reason
Tipo: MQLONG

Código de razón que califica *CompCode*.

Cuando se invoca la salida, contiene el código de razón que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Entre los valores posibles se encuentran MQRC_TRUNCATED_MSG_ACCEPTED, que indica que el mensaje se ha truncado en orden de ajuste en el almacenamiento intermedio proporcionado por la aplicación, y MQRC_NOT_CONVERT, que indica que el mensaje requiere conversión pero que todavía no se ha realizado.

En la salida de la salida, este campo contiene la razón que debe devolverse a la aplicación en el parámetro **Reason** de la llamada MQGET; se recomienda lo siguiente:

- Si *Reason* tenía el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada a la salida, los campos *Reason* y *CompCode* no deben alterarse, independientemente de si la conversión es satisfactoria o falla.

  (Si el campo *CompCode* no es MQCC_OK, la aplicación que recupera el mensaje puede identificar un error de conversión comparando los valores *Encoding* y *CodedCharSetId* devueltos en el descriptor de mensaje con los valores solicitados; por el contrario, la aplicación no puede distinguir un mensaje truncado de un mensaje que encajaba en el almacenamiento intermedio. Por este motivo, MQRC_TRUNCATED_MSG_ACCEPTED debe devolverse con preferencia a cualquiera de las razones que indican un error de conversión.)
- Si *Reason* tenía cualquier otro valor en la entrada para la salida:
  - Si la conversión se realiza correctamente, *CompCode* debe establecerse en MQCC_OK y *Reason* debe establecerse en MQRC_NONE.
  - Si la conversión falla, o el mensaje se expande y se tiene que truncar para que quepa en el almacenamiento intermedio, *CompCode* debe establecerse en MQCC_WARNING (o dejarse sin modificar), y *Reason* establecerse en uno de los valores listados, para indicar la naturaleza de la anomalía.

    Tenga en cuenta que si el mensaje después de la conversión es demasiado grande para el almacenamiento intermedio, sólo se debe truncar si la aplicación que ha emitido la llamada MQGET ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG:

    - Si ha especificado esta opción, se devuelve la razón MQRC_TRUNCATED_MSG_ACCEPTED.
    - Si no ha especificado esa opción, el mensaje se devuelve sin convertir, con el código de razón MQRC_CONVERTED_MSG_TOO_BIG.

Los códigos de razón listados se recomiendan para que los utilice la salida para indicar la razón por la que ha fallado la conversión, pero la salida puede devolver otros valores del conjunto de códigos MQRC_ * si se considera apropiado. Además, el rango de valores de MQRC_APPL_FIRST a MQRC_APPL_LAST se asignan para que los utilice la salida para indicar las condiciones en las que la salida desea comunicarse con la aplicación que emite la llamada MQGET.

Nota: Si el mensaje no se puede convertir correctamente, la salida debe devolver MQXDR_CONVERSION_FAILED en el campo *ExitResponse* , para que el gestor de colas devuelva el mensaje sin convertir. Esto es cierto independientemente del código de razón devuelto en el campo *Reason* .

MQRC_APPL_PRIMERO
(900, X'384 ') Valor más bajo para el código de razón definido por la aplicación.

MQRC_APPL_LAST
(999, X'3E7') Valor más alto para el código de razón definido por la aplicación.

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_NOT_CONVERTED
(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

Es un campo de entrada/salida para la salida.

`StrucId`
Tipo: MQCHAR4

Identificador de estructura.El valor debe ser:

MQDXP_STRUC_ID
Identificador de la estructura de parámetros de salida de conversión de datos.

Para el lenguaje de programación C, también se define la constante MQDXP_STRUC_ID_ARRAY; tiene el mismo valor que MQDXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

`Version`
Tipo: MQLONG

Número de versión de la estructura.El valor debe ser:

MQDXP_VERSION_1
Número de versión para la estructura de parámetros de salida de conversión de datos.

La constante siguiente especifica el número de versión de la versión actual:

MQDXP_CURRENT_VERSION
Versión actual de la estructura de parámetros de salida de conversión de datos.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el campo *`Version`* es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

Declaración C

```
typedef struct tagMQDXP MQDXP;
struct tagMQDXP {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   ExitOptions;     /* Reserved */
  MQLONG   AppOptions;      /* Application options */
  MQLONG   Encoding;        /* Numeric encoding required by
                               application */
  MQLONG   CodedCharSetId;  /* Character set required by application */
  MQLONG   DataLength;      /* Length in bytes of message data */
  MQLONG   CompCode;        /* Completion code */
  MQLONG   Reason;          /* Reason code qualifying CompCode */
  MQLONG   ExitResponse;    /* Response from exit */
  MQHCONN  Hconn;           /* Connection handle */
  PMQIEP   pEntryPoints;    /* Address of the MQIEP structure */
};
```

Declaración COBOL (solo IBM i)

```
**   MQDXP structure
  10 MQDXP.
**    Structure identifier
   15 MQDXP-STRUCID        PIC X(4).
**    Structure version number
   15 MQDXP-VERSION        PIC S9(9) BINARY.
**    Reserved
   15 MQDXP-EXITOPTIONS    PIC S9(9) BINARY.
**    Application options
   15 MQDXP-APPOPTIONS     PIC S9(9) BINARY.
**    Numeric encoding required by application
   15 MQDXP-ENCODING       PIC S9(9) BINARY.
**    Character set required by application
   15 MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**    Length in bytes of message data
   15 MQDXP-DATALENGTH     PIC S9(9) BINARY.
**    Completion code
   15 MQDXP-COMPCODE       PIC S9(9) BINARY.
**    Reason code qualifying COMPCODE
   15 MQDXP-REASON         PIC S9(9) BINARY.
**    Response from exit
   15 MQDXP-EXITRESPONSE   PIC S9(9) BINARY.
**    Connection handle
   15 MQDXP-HCONN          PIC S9(9) BINARY.
```

Declaración de ensamblador System/390

```
MQDXP                 DSECT
MQDXP_STRUCID         DS   CL4  Structure identifier
MQDXP_VERSION         DS   F    Structure version number
MQDXP_EXITOPTIONS     DS   F    Reserved
MQDXP_APPOPTIONS      DS   F    Application options
MQDXP_ENCODING        DS   F    Numeric encoding required by application
MQDXP_CODEDCHARSETID  DS   F    Character set required by application
MQDXP_DATALENGTH      DS   F    Length in bytes of message data
MQDXP_COMPCODE        DS   F    Completion code
MQDXP_REASON          DS   F    Reason code qualifying COMPCODE
MQDXP_EXITRESPONSE    DS   F    Response from exit
MQDXP_HCONN           DS   F    Connection handle
*
MQDXP_LENGTH          EQU  *-MQDXP
                      ORG  MQDXP
MQDXP_AREA            DS   CL(MQDXP_LENGTH)
```

MQXCNVC-Convertir caracteres

La llamada MQXCNVC convierte los caracteres de un juego de caracteres a otro utilizando el lenguaje de programación C.

Esta llamada forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ .

Nota: La llamada se puede utilizar desde los entornos de salida de aplicación y de conversión de datos.

Sintaxis

MQXCNVC (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

En una salida de conversión de datos, Hconn es normalmente el descriptor de contexto que se pasa a la salida de conversión de datos en el campo Hconn de la estructura MQDXP; este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

IBM i En IBM i, se puede especificar el siguiente valor especial para Hconn:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Si ejecuta una aplicación CICS TS 3.2 o superior, asegúrese de que el programa de salida de conversión de caracteres, que invoca la llamada MQXCNVC, esté definido como OPENAPI. Esta definición evita el error 2018 MQRC_HCONN_ERROR provocado por una conexión incorrecta y permite que se complete la MQGET.

Opciones

Tipo: MQLONG - entrada

Opciones que controlan la acción de MQXCNVC.

Se pueden especificar cero o más de las opciones descritas. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

Opción de conversión predeterminada: la siguiente opción controla el uso de la conversión de caracteres predeterminada:

MQDCC_DEFAULT_CONVERSION
Conversión predeterminada.

Esta opción especifica que se puede utilizar la conversión de caracteres por omisión si uno o ambos de los juegos de caracteres especificados en la llamada no están soportados. Esto permite al gestor de colas utilizar un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de caracteres especificado, al convertir la serie.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir la serie es que algunos caracteres se pueden convertir incorrectamente. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres especificado como al juego de caracteres predeterminado.

Los juegos de caracteres predeterminados se definen mediante una opción de configuración cuando se instala o se reinicia el gestor de colas.

Si no se especifica MQDCC_DEFAULT_CONVERSION, el gestor de colas sólo utiliza los juegos de caracteres especificados para convertir la serie, y la llamada falla si uno o ambos juegos de caracteres no están soportados.

Esta opción está soportada en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

Opción de relleno: la opción siguiente permite al gestor de colas rellenar la serie convertida con espacios en blanco o descartar caracteres finales insignificantes, para que la serie convertida se ajuste al almacenamiento intermedio de destino:

MQDCC_FILL_TARGET_BUFFER
Rellene el almacenamiento intermedio de destino.

Esta opción solicita que la conversión tenga lugar de forma que el almacenamiento intermedio de destino se llene completamente:

- Si la serie se contrae cuando se convierte, se añaden espacios en blanco finales para rellenar el almacenamiento intermedio de destino.
- Si la serie se expande cuando se convierte, los caracteres finales que no son significativos se descartan para que la serie convertida se ajuste al almacenamiento intermedio de destino. Si esto se puede realizar correctamente, la llamada se completa con MQCC_OK y el código de razón MQRC_NONE.

  Si hay muy pocos caracteres finales insignificantes, la mayor parte de la serie que cabe se coloca en el almacenamiento intermedio de destino y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG.

  Los caracteres insignificantes son:

  – Blancos de cola
  – Caracteres que siguen al primer carácter nulo de la serie (pero excluyendo el propio primer carácter nulo)
- Si la serie `TargetCCSID`y `TargetLength` son tales que el almacenamiento intermedio de destino no se puede establecer completamente con caracteres válidos, la llamada falla con MQCC_FAILED y el código de razón MQRC_TARGET_LENGTH_ERROR. Esto puede ocurrir cuando `TargetCCSID` es un juego de caracteres DBCS puro (como UTF-16), pero `TargetLength` especifica una longitud que es un número impar de bytes.
- `TargetLength` puede ser menor o mayor que `SourceLength`. Al volver de MQXCNVC, `DataLength` tiene el mismo valor que `TargetLength`.

Si no se especifica esta opción:

- La serie puede contraerse o expandirse dentro del almacenamiento intermedio de destino según sea necesario. Los caracteres finales insignificantes no se añaden ni descartan.

  Si la serie convertida se ajusta al almacenamiento intermedio de destino, la llamada se completa con MQCC_OK y el código de razón MQRC_NONE.

  Si la serie convertida es demasiado grande para el almacenamiento intermedio de destino, la mayor parte de la serie que cabe se coloca en el almacenamiento intermedio de destino y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG. Tenga en cuenta que se pueden devolver menos de `TargetLength` bytes en este caso.
- `TargetLength` puede ser menor o mayor que `SourceLength`. Al volver de MQXCNVC, `DataLength` es menor o igual que `TargetLength`.

Esta opción está soportada en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

Opciones de codificación: Las opciones descritas se pueden utilizar para especificar las codificaciones de enteros de las series de origen y de destino. La codificación relevante sólo se utiliza cuando el identificador de juego de caracteres correspondiente indica que la representación del juego de caracteres en el almacenamiento principal depende de la codificación utilizada para los enteros binarios. Esto sólo afecta a determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16).

La codificación se ignora si el juego de caracteres es un juego de caracteres de un solo byte (SBCS), o un juego de caracteres de varios bytes con representación en el almacenamiento principal que no depende de la codificación de enteros.

Sólo se debe especificar uno de los valores MQDCC_SOURCE_ *, combinado con uno de los valores MQDCC_TARGET_ *:

MQDCC_SOURCE_ENC_NATIVE
La codificación de origen es el valor predeterminado para el entorno y el lenguaje de programación.

MQDCC_SOURCE_ENC_NORMAL
La codificación de origen es normal.

MQDCC_SOURCE_ENC_REVERSE
La codificación de origen se invierte.

MQDCC_SOURCE_ENC_UNDEFINED
La codificación de origen no está definida.

MQDCC_TARGET_ENC_NATIVE
La codificación de destino es el valor predeterminado para el entorno y el lenguaje de programación.

MQDCC_TARGET_ENC_NORMAL
La codificación de destino es normal.

MQDCC_TARGET_ENC_REVERSE
La codificación de destino se invierte.

MQDCC_TARGET_ENC_UNDEFINED
La codificación de destino no está definida.

Los valores de codificación definidos anteriormente se pueden añadir directamente al campo `Options` . Sin embargo, si la codificación de origen o destino se obtiene del campo `Encoding` en el MQMD u otra estructura, se debe realizar el proceso siguiente:

1. La codificación de enteros debe extraerse del campo `Encoding` eliminando las codificaciones decimal flotante y empaquetado; consulte “Análisis de codificaciones” en la página 918 para obtener detalles sobre cómo hacerlo.
2. La codificación de enteros resultante del paso 1 debe multiplicarse por el factor adecuado antes de añadirse al campo `Options` . Estos factores son:
   - MQDCC_SOURCE_ENC_FACTOR para la codificación de origen
   - MQDCC_TARGET_ENC_FACTOR para la codificación de destino

El código de ejemplo siguiente ilustra cómo se puede codificar en el lenguaje de programación C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
```

```
            * MQDCC_SOURCE_ENC_FACTOR
         + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
            * MQDCC_TARGET_ENC_FACTOR;
```

Si no se especifica, las opciones de codificación toman como valor predeterminado sin definir (MQDCC_ * _ENC_UNDEFINED). En la mayoría de los casos, esto no afecta a la finalización satisfactoria de la llamada MQXCNVC. Sin embargo, si el juego de caracteres correspondiente es un juego de caracteres multibyte con representación que depende de la codificación (por ejemplo, un juego de caracteres UTF-16), la llamada falla con el código de razón MQRC_SOURCE_INTEGER_ENC_ERROR o MQRC_TARGET_INTEGER_ENC_ERROR según corresponda.

Las opciones de codificación están soportadas en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows
- z/OS

Opción predeterminada: si no se especifica ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

MQDCC_NONE
No se ha especificado ninguna opción.

MQDCC_NONE está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

SourceCCSID
Tipo: MQLONG - entrada

Es el identificador de juego de caracteres codificado de la serie de entrada en `SourceBuffer`.

SourceLength
Tipo: MQLONG - entrada

Es la longitud en bytes de la serie de entrada en `SourceBuffer` ; debe ser cero o mayor.

SourceBuffer
Tipo: MQCHAR x SourceLength -entrada

Es el almacenamiento intermedio que contiene la serie que se va a convertir de un juego de caracteres a otro.

TargetCCSID
Tipo: MQLONG - entrada

Es el identificador de juego de caracteres codificado del juego de caracteres al que se va a convertir `SourceBuffer` .

TargetLength
Tipo: MQLONG - entrada

Es la longitud en bytes del almacenamiento intermedio de salida `TargetBuffer` ; debe ser cero o mayor. Puede ser menor o mayor que `SourceLength`.

TargetBuffer
Tipo: MQCHAR x TargetLength -salida

Esta es la serie después de que se haya convertido al juego de caracteres definido por `TargetCCSID`. La serie convertida puede ser más corta o más larga que la serie no convertida. El parámetro **DataLength** indica el número de bytes válidos devueltos.

DataLength
Tipo: MQLONG - salida

Es la longitud de la serie devuelta en el almacenamiento intermedio de salida `TargetBuffer`. La serie convertida puede ser más corta o más larga que la serie no convertida.

CompCode
Tipo: MQLONG - salida

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica `CompCode`.

Si `CompCode` es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si `CompCode` es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

Si `CompCode` es MQCC_FAILED:

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DBCS_ERROR
(2150, X'866') La serie DBCS no es válida.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TARGET_LENGTH_ERROR
(2144, X'860 ') El parámetro de longitud de destino no es válido.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Mensajes y códigos de razón.

Invocación en C

```
MQXCNVC (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
        TargetCCSID, TargetLength, TargetBuffer, &DataLength,
        &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;            /* Connection handle */
MQLONG   Options;          /* Options that control the action of
                              MQXCNVC */
MQLONG   SourceCCSID;      /* Coded character set identifier of string
                              before conversion */
MQLONG   SourceLength;     /* Length of string before conversion */
MQCHAR   SourceBuffer[n];  /* String to be converted */
MQLONG   TargetCCSID;      /* Coded character set identifier of string
                              after conversion */
MQLONG   TargetLength;     /* Length of output buffer */
MQCHAR   TargetBuffer[n];  /* String after conversion */
MQLONG   DataLength;       /* Length of output string */
MQLONG   CompCode;         /* Completion code */
MQLONG   Reason;           /* Reason code qualifying CompCode */
```

Declaración COBOL (solo IBM i)

IBM i

```
     CALL 'MQXCNVC' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                          SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                          TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Options that control the action of MQXCNVC
 01  OPTIONS       PIC S9(9) BINARY.
**   Coded character set identifier of string before conversion
 01  SOURCECCSID   PIC S9(9) BINARY.
**   Length of string before conversion
 01  SOURCELENGTH  PIC S9(9) BINARY.
**   String to be converted
 01  SOURCEBUFFER  PIC X(n).
**   Coded character set identifier of string after conversion
 01  TARGETCCSID   PIC S9(9) BINARY.
**   Length of output buffer
 01  TARGETLENGTH  PIC S9(9) BINARY.
**   String after conversion
 01  TARGETBUFFER  PIC X(n).
**   Length of output string
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
```

```
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Declaración de ensamblador S/390

```
         CALL MQXCNVC,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,         X
               SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER,     X
               DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN         DS  F      Connection handle
OPTIONS       DS  F      Options that control the action of MQXCNVC
SOURCECCSID   DS  F      Coded character set identifier of string before
*                        conversion
SOURCELENGTH  DS  F      Length of string before conversion
SOURCEBUFFER  DS  CL(n)  String to be converted
TARGETCCSID   DS  F      Coded character set identifier of string after
*                        conversion
TARGETLENGTH  DS  F      Length of output buffer
TARGETBUFFER  DS  CL(n)  String after conversion
DATALENGTH    DS  F      Length of output string
COMPCODE      DS  F      Completion code
REASON        DS  F      Reason code qualifying COMPCODE
```

MQ_DATA_CONV_EXIT-Salida de conversión de datos

La llamada MQ_DATA_CONV_EXIT describe los parámetros que se pasan a la salida de conversión de datos.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_DATA_CONV_EXIT (consulte la nota de uso 11).

Esta definición forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ .

Sintaxis

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parámetros

DataConvExitParms

Tipo: MQDXP-entrada/salida

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar el resultado de la conversión. Consulte “MQDXP-Parámetro de salida de conversión de datos” en la página 929 para obtener detalles de los campos de esta estructura.

MsgDesc

Tipo: MQMD - entrada/salida

En la entrada a la salida, este es el descriptor de mensaje asociado con los datos de mensaje pasados a la salida en el parámetro **InBuffer** .

Nota: El parámetro **MsgDesc** pasado a la salida es siempre la versión más reciente de MQMD soportada por el gestor de colas que invoca la salida. Si la salida está pensada para ser portable entre distintos entornos, la salida comprobará el campo `Version` en `MsgDesc` para verificar que los campos a los que la salida necesita acceder están presentes en la estructura.

En los entornos siguientes, a la salida se le pasa un MQMD version-2 :

- AIX
- IBM i
- Linux
- Solaris
- Windows

En todos los demás entornos que dan soporte a la salida de conversión de datos, a la salida se le pasa un MQMD de version-1 .

En la salida, la salida cambiará los campos `Encoding` y `CodedCharSetId` por los valores solicitados por la aplicación, si la conversión ha sido satisfactoria; estos cambios se reflejan de nuevo en la aplicación. Cualquier otro cambio que realice la salida en la estructura se ignorará; no se reflejarán de nuevo en la aplicación.

Si la salida devuelve MQXDR_OK en el campo `ExitResponse` de la estructura MQDXP, pero no cambia los campos `Encoding` o `CodedCharSetId` del descriptor de mensaje, el gestor de colas devuelve para esos campos los valores que los campos correspondientes de la estructura MQDXP tenían en la entrada de la salida.

InBufferLongitud

Tipo: MQLONG - entrada

Longitud en bytes de `InBuffer`.

Es la longitud del almacenamiento intermedio de entrada `InBuffer`y especifica el número de bytes que debe procesar la salida. `InBufferLength` es la longitud menor de los datos del mensaje antes de la conversión y la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

InBuffer

Tipo: MQBYTExInBufferLength -entrada

Almacenamiento intermedio que contiene el mensaje no convertido.

Contiene los datos del mensaje antes de la conversión. Si la salida no puede convertir los datos, el gestor de colas devuelve el contenido de este almacenamiento intermedio a la aplicación una vez completada la salida.

Nota: La salida no debe alterar `InBuffer` ; si se altera este parámetro, los resultados no están definidos.

En el lenguaje de programación C, este parámetro se define como un puntero a vacío.

OutBufferLongitud

Tipo: MQLONG - entrada

Longitud en bytes de `OutBuffer`.

Es la longitud del almacenamiento intermedio de salida `OutBuffer`y es la misma que la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

OutBuffer

Tipo: MQBYTExOutBufferLength -salida

Almacenamiento intermedio que contiene el mensaje convertido.

En la salida de la salida, si la conversión ha sido satisfactoria (como indica el valor MQXDR_OK en el campo `ExitResponse` del parámetro **`DataConvExitParms`**), `OutBuffer` contiene los datos de mensaje que se van a entregar a la aplicación, en la representación solicitada. Si la conversión

no ha sido satisfactoria, se ignoran los cambios que la salida ha realizado en este almacenamiento intermedio.

En el lenguaje de programación C, este parámetro se define como un puntero a vacío.

Notas de uso

1. Una salida de conversión de datos es una salida escrita por el usuario que recibe el control durante el proceso de una llamada MQGET. La función realizada por la salida de conversión de datos la define el proveedor de la salida; sin embargo, la salida debe ajustarse a las reglas descritas aquí y en la estructura de parámetros asociada MQDXP.

   El entorno determina los lenguajes de programación que se pueden utilizar para una salida de conversión de datos.
2. La salida sólo se invoca si se cumplen todas las sentencias siguientes:
   - La opción MQGMO_CONVERT se especifica en la llamada MQGET
   - El campo `Format` del descriptor de mensaje no es MQFMT_NONE
   - El mensaje no está ya en la representación necesaria; es decir, uno o ambos `CodedCharSetId` y `Encoding` del mensaje son diferentes del valor especificado por la aplicación en el descriptor de mensaje proporcionado en la llamada MQGET
   - El gestor de colas todavía no ha realizado la conversión correctamente
   - La longitud del almacenamiento intermedio de la aplicación es mayor que cero
   - La longitud de los datos del mensaje es mayor que cero
   - El código de razón hasta ahora durante la operación MQGET es MQRC_NONE o MQRC_TRUNCATED_MSG_ACCEPTED
3. Cuando se está escribiendo una salida, considere la posibilidad de codificar la salida de una forma que le permita convertir los mensajes que se han truncado. Los mensajes truncados pueden aparecer de las siguientes maneras:
   - La aplicación receptora proporciona un almacenamiento intermedio que es menor que el mensaje, pero especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET.

     En este caso, el campo `Reason` del parámetro **DataConvExitParms** en la entrada de la salida tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED.
   - El remitente del mensaje lo ha truncado antes de enviarlo. Esto puede suceder con los mensajes de informe, por ejemplo (consulte “Conversión de mensajes de informe” en la página 929 para obtener más detalles).

     En este caso, el campo `Reason` del parámetro **DataConvExitParms** en la entrada de la salida tiene el valor MQRC_NONE (si la aplicación receptora ha proporcionado un almacenamiento intermedio lo suficientemente grande para el mensaje).

   Por lo tanto, el valor del campo `Reason` en la entrada a la salida no siempre se puede utilizar para decidir si el mensaje se ha truncado.

   La característica distintiva de un mensaje truncado es que la longitud proporcionada a la salida en el parámetro **InBufferLength** es menor que la longitud implícita por el nombre de formato contenido en el campo `Format` del descriptor de mensaje. Por lo tanto, la salida debe comprobar el valor de `InBufferLength` antes de intentar convertir cualquiera de los datos; la salida no debe suponer que se ha proporcionado la cantidad completa de datos implícita en el nombre de formato.

   Si la salida no se ha grabado para convertir mensajes truncados y `InBufferLength` es menor que el valor esperado, la salida devolverá MQXDR_CONVERSION_FAILED en el campo `ExitResponse` del parámetro **DataConvExitParms** , con los campos `CompCode` y `Reason` establecidos en MQCC_WARNING y MQRC_FORMAT_ERROR.

   Si la salida se ha grabado para convertir mensajes truncados, la salida convertirá la mayor cantidad posible de datos (consulte la siguiente nota de uso), teniendo cuidado de no intentar examinar o convertir datos más allá del final de `InBuffer`. Si la conversión se completa correctamente, la

salida dejará el campo Reason en el parámetro **DataConvExitParms** sin modificar. Esto devuelve MQRC_TRUNCATED_MSG_ACCEPTED si el gestor de colas del receptor ha truncado el mensaje y MQRC_NONE si el emisor del mensaje ha truncado el mensaje.

También es posible que un mensaje se expanda durante la conversión, hasta el punto en el que sea mayor que OutBuffer. En este caso, la salida debe decidir si trunca el mensaje; el campo AppOptions del parámetro **DataConvExitParms** indica si la aplicación receptora ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG.

4. Generalmente, todos los datos del mensaje proporcionados a la salida en InBuffer se convierten, o que ninguno de ellos lo es. Sin embargo, se produce una excepción a esto si el mensaje se trunca, ya sea antes de la conversión o durante la conversión; en este caso puede haber un elemento incompleto al final del almacenamiento intermedio (por ejemplo: 1 byte de un carácter de doble byte o 3 bytes de un entero de 4 bytes). En esta situación, considere la posibilidad de omitir el elemento incompleto y establecer los bytes no utilizados en OutBuffer en nulos. Sin embargo, se deben convertir los elementos o caracteres completos dentro de una matriz o serie.
5. Cuando se necesita una salida por primera vez, el gestor de colas intenta cargar un objeto que tiene el mismo nombre que el formato (aparte de las extensiones). El objeto cargado debe contener la salida que procesa los mensajes con ese nombre de formato. Considere la posibilidad de hacer que el nombre de salida y el nombre del objeto que contiene la salida sean idénticos, aunque no todos los entornos lo requieran.
6. Se carga una nueva copia de la salida cuando una aplicación intenta recuperar el primer mensaje que utiliza ese Format desde que la aplicación se conectó al gestor de colas. Para aplicaciones CICS o IMS , esto significa cuando el subsistema CICS o IMS se ha conectado al gestor de colas. Una nueva copia también se puede cargar en otros momentos, si el gestor de colas ha descartado una copia cargada anteriormente. Por este motivo, una salida no debe intentar utilizar el almacenamiento estático para comunicar información de una invocación de la salida a la siguiente-la salida se puede descargar entre las dos invocaciones.
7. Si hay una salida proporcionada por el usuario con el mismo nombre que uno de los formatos incorporados soportados por el gestor de colas, la salida proporcionada por el usuario no sustituye la rutina de conversión incorporada. Las únicas circunstancias en las que se invoca una salida de este tipo son:
   - Si la rutina de conversión incorporada no puede manejar conversiones a o desde los CodedCharSetId o Encoding implicados, o
   - Si la rutina de conversión incorporada no ha podido convertir los datos (por ejemplo, porque hay un campo o carácter que no se puede convertir).
8. El ámbito de la salida depende del entorno. Los nombres de Format deben elegirse para minimizar el riesgo de conflictos con otros formatos. Considere la posibilidad de empezar con caracteres que identifiquen la aplicación que define el nombre de formato.
9. La salida de conversión de datos se ejecuta en un entorno como el del programa que ha emitido la llamada MQGET; el entorno incluye el espacio de direcciones y el perfil de usuario (si procede). El programa podría ser un agente de canal de mensajes que envía mensajes a un gestor de colas de destino que no soporta la conversión de mensajes. La salida no puede comprometer la integridad del gestor de colas, ya que no se ejecuta en el entorno del gestor de colas.
10. La única llamada MQI que puede utilizar la salida es MQXCNVC; el intento de utilizar otras llamadas MQI falla con el código de razón MQRC_CALL_IN_PROGRESS, u otros errores imprevisibles.
11. El gestor de colas no proporciona ningún punto de entrada denominado MQ_DATA_CONV_EXIT. Sin embargo, se proporciona un typedef para el nombre MQ_DATA_CONV_EXIT en el lenguaje de programación C, y esto se puede utilizar para declarar la salida escrita por el usuario, para asegurarse de que los parámetros son correctos. El nombre de la salida debe ser el mismo que el nombre de formato (el nombre contenido en el campo Format en MQMD), aunque esto no es necesario en todos los entornos.

El ejemplo siguiente ilustra cómo se puede declarar la salida que procesa el formato MYFORMAT en el lenguaje de programación C:

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
     PMQDXP   pDataConvExitParms, /* Data-conversion exit parameter
                                     block */
     PMQMD    pMsgDesc,           /* Message descriptor */
     MQLONG   InBufferLength,     /* Length in bytes of InBuffer */
     PMQVOID  pInBuffer,          /* Buffer containing the unconverted
                                     message */
     MQLONG   OutBufferLength,    /* Length in bytes of OutBuffer */
     PMQVOID  pOutBuffer)         /* Buffer containing the converted
                                     message */
{
  /* C language statements to convert message */
}
```

12. z/OS En z/OS, si también está en vigor una salida cruzada de API, se llama después de la salida de conversión de datos.

Invocación en C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
         InBuffer, OutBufferLength, OutBuffer);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQDXP   DataConvExitParms;  /* Data-conversion exit parameter block */
MQMD    MsgDesc;            /* Message descriptor */
MQLONG  InBufferLength;     /* Length in bytes of InBuffer */
MQBYTE  InBuffer[n];        /* Buffer containing the unconverted
                               message */
MQLONG  OutBufferLength;    /* Length in bytes of OutBuffer */
MQBYTE  OutBuffer[n];       /* Buffer containing the converted
                               message */
```

Declaración COBOL (soloIBM i)

IBM i

```
     CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                           INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   Data-conversion exit parameter block
 01  DATACONVEXITPARMS.
     COPY CMQDXPV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Length in bytes of INBUFFER
 01  INBUFFERLENGTH     PIC S9(9) BINARY.
**   Buffer containing the unconverted message
 01  INBUFFER           PIC X(n).
**   Length in bytes of OUTBUFFER
 01  OUTBUFFERLENGTH    PIC S9(9) BINARY.
**   Buffer containing the converted message
 01  OUTBUFFER          PIC X(n).
```

Declaración de ensamblador System/390

```
         CALL EXITNAME,(DATACONVEXITPARMS,MSGDESC,INBUFFERLENGTH,      X
               INBUFFER,OUTBUFFERLENGTH,OUTBUFFER)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
DATACONVEXITPARMS  CMQDXPA  ,      Data-conversion exit parameter block
MSGDESC            CMQMDA   ,      Message descriptor
INBUFFERLENGTH     DS       F      Length in bytes of INBUFFER
INBUFFER           DS       CL(n)  Buffer containing the unconverted
*                                  message
OUTBUFFERLENGTH    DS       F      Length in bytes of OUTBUFFER
OUTBUFFER          DS       CL(n)  Buffer containing the converted
*                                  message
```

Propiedades especificadas como elementos MQRFH2

Las propiedades de descriptor que no son de mensaje se pueden especificar como elementos en las carpetas de cabecera MQRFH2 . Visión general de los elementos MQRFH2 que se especifican como propiedades.

Esto mantiene la compatibilidad con las versiones anteriores de los clientes IBM MQ JMS y XMS . En esta sección se describe cómo especificar propiedades en las cabeceras MQRFH2 .

Para utilizar elementos MQRFH2 como propiedades, especifique los elementos tal como se describe en Utilización de IBM MQ classes for Java . Esta información complementa la información descrita en “MQRFH2 – Reglas y formato de la cabecera 2” en la página 533.

Correlación de tipos de datos de propiedad con tipos de datos MQRFH2

Este tema proporciona información sobre los tipos de propiedad de mensaje correlacionados con sus tipos de datos MQRFH2 correspondientes.

Tabla 635. Tipos de datos MQRFH2 soportados

Tipo de propiedad de mensaje	Tipo de datos MQRFH2
MQBYTE []	bin.hex
MQBOOL	boolean
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	serie

Se supone que cualquier elemento sin un tipo de datos es de tipo "serie".

Un tipo de datos MQRFH2 de `int`, que significa un entero de tamaño no especificado, se trata como si fuera un `i8`.

Un valor nulo se indica mediante el atributo de elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos.

Por ejemplo, la propiedad siguiente tiene un valor nulo:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Una propiedad de serie de bytes o caracteres puede tener un valor vacío. Esto se representa mediante un elemento MQRFH2 con un valor de elemento de longitud cero.

Por ejemplo, la propiedad siguiente tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Carpetas MQRFH2 soportadas

Visión general del uso de campos de descriptor de mensaje como propiedades.

Las carpetas `<jms>`, `<mcd>`, `<mqext>`y `<usr>` se describen en La cabecera MQRFH2 y JMS. La carpeta `<usr>` se utiliza para transportar las propiedades definidas por la aplicación JMS que están asociadas con un mensaje. Los grupos no están permitidos en la carpeta `<usr>` .

La cabecera MQRFH2 y JMS dan soporte a las siguientes carpetas adicionales:

- `<mq>`

  Esta carpeta se utiliza y se reserva para las propiedades definidas por MQque utiliza IBM MQ.
- `<mq_usr>`

  Esta carpeta se puede utilizar para transportar cualquier propiedad definida por la aplicación que no esté expuesta como propiedades definidas por el usuario JMS , ya que es posible que las propiedades no cumplan los requisitos de una propiedad JMS . Esta carpeta puede contener grupos que la carpeta `<usr>` no puede.
- Cualquier carpeta marcada con el atributo `content='properties'` .

  Una carpeta de este tipo es equivalente a la carpeta `<mq_usr>` en el contenido.
- `<mqps>`

  Esta carpeta se utiliza para las propiedades de publicación/suscripción de IBM MQ .

IBM MQ también da soporte a las carpetas siguientes que ya están en uso por WAS/SIB:

- `<sib>`

  Esta carpeta se utiliza y se reserva para las propiedades de mensajes del sistema WAS/SIB que no están expuestas como propiedades JMS , o que están correlacionadas con propiedades JMS_IBM_ *, pero que están expuestas a aplicaciones WAS/SIB; estas incluyen las propiedades de vías de acceso de direccionamiento inverso y de reenvío.

  Al menos algunos no se pueden exponer como propiedades JMS , porque son matrices de bytes. Si la aplicación añade propiedades a esta carpeta, el valor se ignora o se elimina.
- `<sib_usr>`

  Esta carpeta se utiliza y se reserva para las propiedades de mensaje de usuario WAS/SIB que no se pueden exponer como propiedades de usuario JMS porque no son de tipos soportados; se exponen a aplicaciones WAS/SIB.

  Estas son propiedades de usuario, que puede obtener o establecer a través de la interfaz SIMessage, pero el contenido de la matriz de bytes se correlaciona con el valor de propiedad necesario.

  Si la aplicación IBM MQ escribe un elemento `bin.hex` arbitrario en la carpeta, la aplicación probablemente recibirá un `IOException`, ya que no tiene el formato que se espera restaurar. Si añade algo que no sea un elemento `bin.hex` , recibirá un `ClassCastException`.

  No intente que las propiedades estén disponibles para WAS/SIB utilizando esta carpeta; en su lugar, utilice la carpeta `<usr>` para este fin.
- `<sib_context>`

Esta carpeta se utiliza para las propiedades de mensaje del sistema WAS/SIB que no están expuestas a las aplicaciones de usuario WAS/SIB o como propiedades JMS . Estas incluyen propiedades de seguridad y transaccionales que se utilizan para servicios web y similares.

La aplicación no debe añadir propiedades a esta carpeta.

- `<mqema>`

  WAS/SIB ha utilizado esta carpeta en lugar de la carpeta `<mqext>` .

Los nombres de carpeta MQRFH2 distinguen entre mayúsculas y minúsculas.

Las carpetas siguientes están reservadas, en cualquier combinación de caracteres en minúsculas o mayúsculas:

- Cualquier carpeta con el prefijo `mq` o `wmq` ; reservado para su uso por parte de IBM MQ.
- Cualquier carpeta con el prefijo `sib` ; reservado para su uso por WAS/SIB.
- Carpetas `<Root>` y `<Body>` ; reservadas pero no utilizadas.

Las carpetas siguientes no se reconocen como que contienen propiedades de mensaje:

- `<psc>`

  Lo utiliza IBM Integration Bus para transmitir mensajes de mandatos de publicación/suscripción al intermediario.

- `<pscr>`

  Utilizado por IBM Integration Bus para contener información del intermediario, en respuesta a mensajes de mandato de publicación/suscripción.

- Cualquier carpeta no definida por IBM, que no esté marcada con el atributo `content='properties'` .

No especifique `content='properties'` en las carpetas `<psc>` o `<pscr>` . Si lo hace, estas carpetas se tratan como propiedades y es probable que IBM Integration Bus deje de funcionar como se esperaba.

Si la aplicación está creando mensajes con propiedades, en las cabeceras MQRFH2 que se van a reconocer como una cabecera MQRFH2 que contiene propiedades, la cabecera debe estar en la lista de cabeceras que se pueden encadenar en la cabecera del mensaje.

La MQRFH2 puede ir precedida de cualquier número de cabeceras estándar MQH, o una MQCIH, una MQDLH, una MQIIH, una MQTM, una MQTMC2o una MQXQH. Una serie o un MQCFH finaliza el análisis porque no se pueden encadenar.

Es posible que un mensaje contenga varias cabeceras MQRFH2 que transporten todas las propiedades del mensaje. Las carpetas con el mismo nombre pueden coexistir en cabeceras diferentes a menos que se restrinja lo contrario, por ejemplo, WAS/SIB. Las carpetas se tratan como una carpeta lógica, si todas están en cabeceras significativas.

Mientras que las carpetas de las cabeceras significativas no se pueden fusionar con esas carpetas en cabeceras no significativas, las carpetas con el mismo nombre dentro de las cabeceras significativas se pueden fusionar, eliminando las propiedades en conflicto. Las aplicaciones no deben depender del diseño de las propiedades dentro de su mensaje.

Los grupos MQRFH2 se analizan para ver las propiedades de las carpetas definidas por el usuario, es decir, no las carpetas `<wmq>`, `<jms>`, `<mcd>`, `<usr>`, `<mqext>`, `<sib>`, `<sib_usr>`, `<sib_context>`y `<mqema>` .

Los grupos de las carpetas de propiedades definidas por IBM, excepto las carpetas `<wmq>` y `<mq>` , se analizan para las propiedades.

Una carpeta MQRFH2 no puede contener contenido mixto; una carpeta o grupo puede contener grupos o propiedades, o un valor, pero no ambos.

Un segmento de un mensaje, ya sea el primero o un segmento posterior, no puede contener propiedades definidas por IBM MQ que no sean las propiedades del descriptor de mensaje. Por lo tanto, colocar un mensaje que contenga dichas propiedades con el

conjunto MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED hace que la colocación falle con MQRC_SEGMENTATION_NOT_ALLOWED.

Sin embargo, los grupos de mensajes pueden contener propiedades definidas por IBM MQ .

Generación de cabeceras MQRFH2

Si IBM MQ convierte las propiedades de mensaje a su representación MQRFH2 , debe añadir MQRFH2 al mensaje. Añade MQRFH2 como una cabecera separada o lo fusiona con una cabecera existente.

La generación de nuevas cabeceras MQRFH2 mediante IBM MQ puede interrumpir las cabeceras existentes en un mensaje. Las aplicaciones que analizan un almacenamiento intermedio de mensajes para cabeceras deben tener en cuenta que el número y la posición de las cabeceras en un almacenamiento intermedio pueden cambiar en algunas circunstancias. IBM MQ intenta minimizar el impacto de añadir propiedades a un mensaje fusionando propiedades de mensaje en una cabecera MQRFH2 existente, donde puede. También intenta minimizar el impacto insertando un MQRFH2 generado en una posición fija relativa a otras cabeceras en el almacenamiento intermedio de mensajes.

Una cabecera MQRFH2 generada se coloca a continuación de la MQMDy de cualquier número de cabeceras MQXQH, MQRFHy MQDLH , independientemente del orden en el que se encuentren. La cabecera MQRFH2 generada se coloca inmediatamente antes de la primera cabecera que no es una cabecera MQMD, MQXQH, MQDLHo MQRFH .

z/OS En sistemas z/OS , la cabecera MQRFH2 generada se crea en CCSIDde la aplicación. Esto se define de la siguiente manera:

- Para las aplicaciones LE por lotes que utilizan la interfaz DLL, CCSID es el CODESET asociado con el entorno local actual en el momento en que se emite **MQCONN** (el valor predeterminado es 1047).
- Para las aplicaciones LE por lotes enlazadas con uno de los apéndices de MQ por lotes, CCSID es el CODESET asociado con el entorno local actual en el momento de la primera llamada MQI emitida después de **MQCONN** (el valor predeterminado es 1047).
- Para aplicaciones no LE por lotes que se ejecutan en una hebra USS, CCSID es el valor de THLICCSID en el momento de la primera llamada MQI emitida después de **MQCONN** (el valor predeterminado es 1047).
- Para otras aplicaciones por lotes, CCSID es el CCSID del gestor de colas.

Para las aplicaciones LE, el entorno local se puede cambiar utilizando el servicio invocable `setlocale() / CEESETL LE` . Para aplicaciones no LE que se ejecutan en hebras USS, el valor de THLICCSID se puede cambiar utilizando la macro de correlación USS **BPXYTHLI**.

Reglas para fusionar MQRFH2 generado

Las reglas siguientes se aplican a la fusión de un MQRFH2 generado con un MQRFH2existente. La cabecera MQRFH2 generada se fusiona con una cabecera MQRFH2 existente, si:

1. El MQRFH2 existente está en la misma posición que IBM MQ colocaría un MQRFH2generado o anterior en la cadena de cabecera.
2. El CCSID de las propiedades generadas es el mismo que el NameValueCCSID del MQRFH2existente.

De lo contrario, la cabecera generada se coloca por separado en el almacenamiento intermedio, en la posición descrita anteriormente.

Reglas para fusionar carpetas en una MQRFH2 existente

Si las propiedades de mensaje se fusionan en un MQRFH2existente, el MQRFH2 existente se explora en busca de carpetas que coincidan con las propiedades de mensaje y las fusiona. Si no existe una carpeta coincidente, se añade una carpeta nueva al final de las carpetas existentes. Si existe una carpeta coincidente, se busca en la carpeta. Las propiedades coincidentes se sobrescriben. Las nuevas se añaden al final de la carpeta.

Restricciones de la carpeta MQRFH2

Visión general de las restricciones de carpeta en cabeceras MQRFH2

Las restricciones MQRFH2 se aplican a las carpetas siguientes:

- Los nombres de elemento de la carpeta `<usr>` no deben empezar con el prefijo `JMS` ; dichos nombres de propiedad están reservados para que los utilice JMS y no son válidos para las propiedades definidas por el usuario.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensaje de IBM MQ .
- Los nombres de elemento de la carpeta `<usr>` no pueden ser, en cualquier combinación de minúsculas o mayúsculas, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS y ESCAPE. Estos nombres coinciden con las palabras clave SQL y dificultan el análisis de los selectores, porque `<usr>` es la carpeta predeterminada que se utiliza cuando no se especifica ninguna carpeta para una propiedad determinada en un selector.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensaje de IBM MQ .
- El modelo de contenido de la carpeta `<usr>` es el siguiente:
  - Se puede utilizar cualquier nombre XML válido como nombre de elemento, siempre que no contenga dos puntos.
  - Sólo se permiten elementos simples, no carpetas anidadas.
  - Todos los elementos toman el tipo predeterminado de serie, a menos que lo modifique un atributo `dt="xxx"` .
  - Todos los elementos son opcionales, pero no deben aparecer más de una vez en una carpeta.
- Los nombres de elemento de cualquier carpeta que se considere que contiene propiedades de mensaje no deben contener un punto (.) (Carácter Unicode U+002E), porque se utiliza en los nombres de propiedad para indicar la jerarquía.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensaje de IBM MQ .

En general, IBM MQ puede analizar las cabeceras MQRFH2 que contienen datos de estilo XML válidos sin errores, aunque determinados elementos de la MQRFH2 no son accesibles a través de las API de propiedad de mensaje IBM MQ .

Conflictos de nombre de elemento MQRFH2

Visión general de los conflictos dentro de los nombres de elemento MQRFH2 .

Sólo se puede adjuntar un valor a una propiedad de mensaje. Si un intento de acceder a una propiedad da lugar a un conflicto de valores, se elige uno con preferencia sobre otro.

La sintaxis IBM MQ para acceder a los elementos MQRFH2 permite la identificación exclusiva de un elemento, si una carpeta no contiene elementos con el mismo nombre. Si una carpeta contiene más de un elemento con el mismo nombre, el valor de la propiedad utilizada es el más cercano a la cabecera del mensaje.

Esto se aplica si dos o más carpetas con el mismo nombre están contenidas en diferentes cabeceras MQRFH2 significativas dentro del mismo mensaje.

Puede producirse un conflicto cuando la llamada MQGET se procesa después de que se haya establecido una propiedad de descriptor que no es de mensaje dos veces: a través de una llamada MQSETMP y directamente en la cabecera MQRFH2 en bruto.

Si esto sucede, la propiedad asociada al mensaje por una llamada de API tiene preferencia sobre una en los datos del mensaje, es decir, la de la cabecera MQRFH2 sin formato. Si se produce un conflicto, se considera que se produce lógicamente antes de los datos del mensaje.

Correlación de nombres de propiedad con nombres de carpeta y elemento MQRFH2

Visión general de las diferencias entre los nombres de propiedad y los nombres de elemento en la cabecera MQRFH2 .

Cuando se utiliza cualquiera de las API definidas que finalmente generan cabeceras MQRFH2 , para especificar propiedades de mensaje (por ejemplo, MQ JMS), el nombre de propiedad no es necesariamente el nombre de elemento en la carpeta MQRFH2 .

Por lo tanto, se produce una correlación entre el nombre de propiedad y el elemento MQRFH2 , y de forma inversa, teniendo en cuenta tanto el nombre de carpeta que contiene el elemento como el nombre de elemento. Algunos ejemplos de IBM MQ classes for JMS ya están documentados en Utilización de IBM MQ classes for Java.

Tabla 636. Nombres de propiedad correlacionados con la carpeta MQRFH2 y nombres de elemento

Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de elemento MQRFH2
`JMSDestination`	`jms`	`Dst`
`JMSType`	`mcd`	`Type`, `Set`, `Fmt`
`xxx` (definido por el usuario, donde `xxx` no empieza por JMS)	`usr`	`xxx`

Por lo tanto, cuando una aplicación JMS accede a la propiedad `JMSDestination` , esta se correlaciona con el elemento `Dst` de la carpeta `<jms>` .

Al especificar propiedades como elementos MQRFH2 , IBM MQ define sus elementos de la forma siguiente:

Tabla 637. Nombres de propiedad correlacionados con nombres de carpeta, grupo y elemento MQRFH2

Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de grupo MQRFH2	Nombre de elemento MQRFH2
`<Property>`	`<usr>`	n/d	`<Property>`
`<folder>.<Property>`	`<folder>`	n/d	`<Property>`
`<folder>.<group>.<Property>`	`<folder>`	`<group>`	`<Property>`

Por ejemplo, cuando una aplicación IBM MQ intenta acceder a la propiedad `Property1` , se correlaciona con el elemento `Property1` en la carpeta `<usr>` . La propiedad `wmq.Property2` se correlaciona con la propiedad `Property2` en la carpeta `<wmq>` .

Si el nombre de propiedad contiene más de uno. , el nombre de elemento MQRFH2 utilizado es el que sigue al final. y los grupos MQRFH2 se utilizan para formar una jerarquía; se permiten grupos MQRFH2 anidados.

Una aplicación IBM MQ accede a la cabecera JMS y a las propiedades específicas del proveedor contenidas en una MQRFH2 en las carpetas `<mcd>`, `<jms>`y `<mqext>` utilizando los nombres abreviados definidos en Utilización de IBM MQ classes for Java .

Se accede a las propiedades definidas por el usuario de JMS desde la carpeta `<usr>` . Una aplicación IBM MQ puede utilizar la carpeta `<usr>` para sus propiedades de aplicación si es aceptable que la propiedad aparezca en las aplicaciones JMS como una de sus propiedades definidas por el usuario.

Si no es aceptable, elija otra carpeta; la carpeta `<wmq_usr>` se proporciona como ubicación estándar para dichas propiedades noJMS .

Las aplicaciones pueden especificar y utilizar cualquier carpeta MQRFH2 con un uso bien definido, no documentado en “Propiedades especificadas como elementos MQRFH2” en la página 947 si observa lo siguiente:

1. Es posible que la carpeta ya esté en uso, o que la utilice en el futuro, otra aplicación que proporcione acceso no definido a las propiedades contenidas en ella; consulte Nombres de propiedad para ver el convenio de denominación sugerido para los nombres de propiedad.
2. Las propiedades no son accesibles para las versiones anteriores del cliente IBM MQ classes for JMS o XMS que sólo puede acceder a la carpeta `<usr>` para las propiedades definidas por el usuario
3. La carpeta debe estar marcada con el atributo `content` con el valor establecido en `properties`, por ejemplo, `content='properties'`.

   "MQSETMP-Establecer propiedad de mensaje" en la página 791 añade automáticamente este atributo según sea necesario. Este atributo no se debe añadir a ninguna de las carpetas definidas por IBM, por ejemplo, `<jms>` y `<usr>`. De este modo, hace que el cliente IBM MQ classes for JMS rechace el mensaje antes de IBM WebSphere MQ 7.0. con un `MessageFormatException`.

Puesto que la carpeta `<usr>` es la ubicación predeterminada para las propiedades de la sintaxis `<Property>` , una aplicación IBM MQ y una aplicación JMS para acceder al mismo valor de propiedad definido por el usuario utilizando el mismo nombre.

Nombres de carpeta reservados

Hay varios nombres de carpeta reservados. No puede utilizar nombres como los prefijos de carpeta; por ejemplo, `Root.Property1` no accede a una propiedad válida porque `Root` está reservado. La lista siguiente contiene nombres de carpeta reservados:

- Raíz
- Body (Cuerpo)
- Propiedades
- Entorno
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

Correlación de campos de descriptor de propiedad en cabeceras MQRFH2

Cuando una propiedad se convierte en un elemento MQRFH2 , se utilizan los siguientes atributos de elemento para especificar los campos significativos del descriptor de propiedad: describe cómo se convierten los campos MQPD en atributos del elemento MQRFH2 .

Soporte

El campo Descriptor de propiedad de soporte se divide en tres atributos de elemento

- El atributo de elemento **sr** especifica valores en la máscara de bits MQPD_REJECT_UNSUP_MASK.
- El atributo de elemento **sa** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_MASK.

- El atributo de elemento **sx** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Estos atributos de elemento sólo son válidos en la carpeta < mq> y se ignoran si se establecen en elementos de las otras carpetas que contienen propiedades.

Tabla 638. Campos MQPD correlacionados con atributos de elemento MQRFH2

Valor de soporte	Atributo de elemento MQRFH2	Valor de atributo MQRFH2
MQPD_SUPPORT_OPTIONAL	sa	opcional Este es el valor predeterminado.
MQPD_SUPPORT_REQUIRED	sr	necesario
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	locales

Contexto

Utilice el atributo de elemento **`context`** para indicar el contexto de mensaje al que pertenece una propiedad. Utilice un solo valor. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Tabla 639. Valores de contexto correlacionados con valores de atributo MQRFH2

Valor de contexto	Valor de atributo MQRFH2
MQPD_NO_CONTEXT	ninguno Este es el valor predeterminado.
MQPD_USER_CONTEXT	usuario

CopyOptions

Utilice el atributo de elemento **`copy`** para indicar los mensajes en los que se debe copiar una propiedad. Se acepta más de un valor; separe varios valores con una coma. Por ejemplo, **`copy='reply'`** y **`copy='publish,report'`** son válidos. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Nota: En la definición de atributo, las comillas simples o las comillas dobles son un uso válido, por ejemplo **`copy='reply'`** o **`copy="report"`**

Tabla 640. Valores de CopyOption correlacionados con valores de atributo MQRFH2

Valor de CopyOption	Valor de atributo MQRFH2
MQPD_COPY_FORWARD	hacia adelante
MQPD_COPY_REPLY	responder
MQPD_COPIA_INFORME	informe
MQPD_COPY_PUBLISH	publicar
MQPD_COPY_ALL	Todo No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, tiene prioridad sobre cualquier valor excepto **none**.

Tabla 640. Valores de CopyOption correlacionados con valores de atributo MQRFH2 (continuación)	
Valor de CopyOption	**Valor de atributo MQRFH2**
MQPD_COPY_DEFAULT	predeterminado Este es el valor predeterminado. Es equivalente a especificar los tres valores MQCOPY_FORWARD, MQCOPY_REPORT y MQCOPY_PUBLISH. No especifique esto con ningún otro valor.
MQPD_COPY_NONE	ninguno No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, esto tiene prioridad.

Restricciones a la carpeta < mq> MQRFH2

Cuando se coloca un mensaje en una cola, se busca una carpeta < mq> para que el mensaje se pueda procesar de acuerdo con sus propiedades definidas por MQ. Para permitir el análisis eficaz de las propiedades definidas por MQ, se aplican las restricciones siguientes a la carpeta:

- MQsólo actúa sobre las propiedades de la primera carpeta < mq> significativa del mensaje; las propiedades de cualquier otra carpeta < mq> del mensaje se ignoran.
- Si la carpeta está en UTF-8, sólo se permiten caracteres UTF-8 de un solo byte en la carpeta. Un carácter de varios bytes en la carpeta puede hacer que falle el análisis y que se rechace el mensaje.
- No incluya grupos MQRFH2 en la carpeta < mq>. La presencia del carácter Unicode U+003C en un valor de propiedad hará que se rechace el mensaje.
- No utilice series de escape en la carpeta. Una serie de escape se trata como el valor real del elemento.
- Sólo el carácter Unicode U+0020 se trata como un espacio en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden hacer que falle el análisis de la carpeta y que se rechace el mensaje.

Si el análisis de la carpeta < mq> falla, o si la carpeta no observa estas restricciones, el mensaje se rechaza con CompCode **MQCC_FAILED** y Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Cabeceras MQRFH2 que no son válidas

En el momento en que se procesa una llamada MQPUT, MQPUT1o MQGET, se puede producir un análisis parcial de las cabeceras MQRFH2 del mensaje para comprobar qué carpetas se incluyen y para determinar si las carpetas contienen propiedades. Visión general de cabeceras MQRFH2 que no son válidas.

Si el análisis parcial del mensaje no se puede completar correctamente porque la estructura no es válida, por ejemplo, el campo `StrucLength` es demasiado pequeño, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_ERROR, si se puede determinar que la aplicación incluye alguna opción IBM WebSphere MQ 7 , para que las aplicaciones existentes no fallen.
- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.

Si el análisis parcial falla porque no se puede detectar si una carpeta determinada contiene propiedades o no, por ejemplo, la carpeta empieza `<<jms`, por lo que el análisis falla antes de que se determine el nombre de la carpeta, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_FORMAT_ERROR, si se puede determinar que la aplicación incluye alguna opción IBM WebSphere MQ 7 , para que las aplicaciones existentes no fallen.

- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.
- Mientras está internamente dentro del gestor de colas, el mensaje no se rechaza debido a la carpeta con formato incorrecto, pero la carpeta siempre se trata como si no hubiera ninguna propiedad dentro de ella.

Un mensaje puede fluir a través de la red de gestores de colas con una carpeta que contenga un error de sintaxis de este tipo, pero que nunca se analice y se detecte, mientras que una o más carpetas del mensaje son:

- Válido
- Se ha analizado correctamente
- Se utiliza en el proceso del mensaje

Por lo tanto, la detección no está garantizada.

Si una de las aplicaciones utiliza "MQSETMP-Establecer propiedad de mensaje" en la página 791, o MQINQMP para acceder a una propiedad, y al hacerlo hace que una carpeta MQRFH2 se analice completamente, detectando un error de forma que el análisis no se puede completar, esto se indica mediante un código de retorno adecuado a la llamada de API. No se pone a disposición de la aplicación ninguna propiedad de la carpeta.

Si se intenta analizar completamente una carpeta MQRFH2 y el analizador encuentra atributos de elemento no reconocidos, o un tipo de datos no reconocido, el análisis continúa y se completa correctamente sin que se emitan avisos; esto no constituye un error de análisis.

Conversión de páginas de códigos

En esta sección se describen los nombres de conjuntos de códigos y los CCSID, el idioma nacional, la conversión de z/OS , la conversión de IBM i , y el soporte de conversión Unicode.

Cada sección de idioma nacional lista la siguiente información:

- Los CCSID nativos soportados
- Las conversiones de página de códigos que no están soportadas

En la información se utilizan los términos siguientes:

AIX **AIX**
Indica IBM MQ for AIX.

Linux **Linux**
Indica IBM MQ para Linux para Intel y IBM MQ para Linux para zSeries.

IBM i **OS/400**
Indica IBM MQ for IBM i.

Solaris **Solaris**
Indica IBM MQ for Solaris.

Windows **Windows**
Indica IBM MQ for Windows.

z/OS **z/OS**
Indica IBM MQ for z/OS.

El valor predeterminado para la conversión de datos es que la conversión se realice en el sistema de destino (receptor).

Si el producto de origen da soporte a la conversión, se puede configurar un canal y se pueden intercambiar datos estableciendo el atributo de canal CONVERT en YES en el origen.

Nota:

1. La conversión para la información de IBM MQ MQI client tiene lugar en el servidor, por lo que el servidor debe dar soporte a la conversión del CCSID del cliente al CCSID del servidor.
2. La conversión puede incluir soporte añadido por CSD/PTF a la última versión de IBM MQ. Compruebe el contenido del nivel de servicio más reciente para ver si necesita instalar un CSD/PTF para habilitar esta conversión.
3. El CCSID del gestor de colas IBM MQ debe ser mixto o SBCS.
4. Algunos CCSID, por ejemplo 850 en AIX, que no están soportados por el sistema operativo pueden seguir siendo utilizados por la aplicación y también pueden establecerse como CCSID del gestor de colas IBM MQ . Esto sólo se permite con fines de compatibilidad con versiones anteriores y la conversión fallará si las tablas de conversión relevantes no están instaladas.

Consulte Tabla 641 en la página 957 para obtener una referencia cruzada entre algunos de los números CCSID y algunos nombres de conjuntos de códigos de la industria.

Referencia relacionada

"Idiomas nacionales" en la página 958
Esta información contiene idiomas soportados por IBM MQ.

Nombres de conjuntos de códigos y CCSID

Nombres de conjunto de códigos y los CCSID correspondientes para cada nombre de conjunto de códigos.

z/OS IBM MQ for z/OS proporciona más conversión de la que se lista en las tablas específicas de idioma. Para obtener una lista completa de conversiones, consulte Tabla 674 en la página 985.

Tabla 641. Nombres de conjuntos de códigos y CCSID

Nombres de conjunto de códigos	CCSID
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (euros)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
pck	943
GBK	1386
koi8-r	878

Idiomas nacionales

Esta información contiene idiomas soportados por IBM MQ.

Los idiomas soportados por IBM MQ son:

- Inglés de EE.UU.-consulte el tema “Inglés norteamericano” en la página 958
- Alemán-consulte el tema “Alemán” en la página 959
- Danés y noruego-véase el tema “Danés y noruego” en la página 960
- Finés y sueco-véase el tema “Finés y sueco” en la página 961
- Italiano-consulte el tema “Italiano” en la página 962
- Español-consulte el tema “Español” en la página 962
- Inglés/Gaélico del Reino Unido-consulte el tema “Inglés del Reino Unido /Gaélico” en la página 963
- Francés-consulte el tema “Francés” en la página 963
- Multilingüe-consulte el tema “Multi idioma” en la página 964
- Portugués-consulte el tema “Portugués” en la página 965
- Islandés-consulte el tema “Islandés” en la página 966
- Lenguas de Europa Oriental-véase el tema “Lenguas de Europa oriental” en la página 966
- Cirílico-consulte el tema “Cirílico” en la página 968
- Estonio-consulte el tema “Estonio” en la página 968
- Letón y lituano-véase el tema “Letón y lituano” en la página 970
- Ucraniano-consulte el tema “Ucraniano” en la página 971
- Griego-consulte el tema “Griego” en la página 971
- Turco-consulte el tema “Turco” en la página 972
- Hebreo-consulte el tema “Hebreo” en la página 973
- Farsi-consulte el tema “Farsi” en la página 975
- Urdu-consulte el tema “Urdu” en la página 975
- Tailandés-consulte el tema “Tailandés” en la página 976
- Lao-consulte el tema “Lao” en la página 976
- Vietnamita-consulte el tema “Vietnamita” en la página 977
- Japonés latín SBCS-consulte el tema “Japonés latín SBCS” en la página 977
- Japonés Katakana SBCS-consulte el tema “Japonés Katakana SBCS” en la página 979
- Japonés Kanji/latín mixto-consulte el tema “Japonés Kanji/latín mixto” en la página 980
- Japonés Kanji/Katakana mixto-consulte el tema “Japonés Kanji/Katakana mixto” en la página 981
- Coreano-consulte el tema “Coreano” en la página 983
- Chino simplificado-consulte el tema “Chino simplificado” en la página 983
- Chino tradicional-consulte el tema “Chino tradicional” en la página 984

Inglés norteamericano

Detalles de CCSID y conversión de CCSID para inglés de EE.UU.

Tabla 642. CCSID nativos para inglés de EE.UU. en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	37, 924, 1140

Tabla 642. CCSID nativos para inglés de EE.UU. en plataformas soportadas (continuación)	
Plataforma	**CCSID nativos**
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 1252, 5348, 858
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

37
No se convierte a las páginas de códigos 923, 858

924
No se convierte a las páginas de códigos 437, 858, 1051, 1140, 1252, 1275, 5348

1140
No se convierte a las páginas de códigos 924, 1051, 1275

Alemán

Detalles de CCSID y conversión de CCSID para alemán.

Tabla 643. CCSID nativos para alemán en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	273, 924, 1141
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

273
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141
No se convierte a las páginas de códigos 924, 1051, 1275

Danés y noruego

Detalles de CCSID y conversión de CCSID para danés y noruego.

Tabla 644. CCSID nativos para danés y noruego en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	277, 924, 1142
AIX AIX	819, 923, 5348
Windows Windows	850, 858, 865, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

277
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No convierte en páginas de códigos 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142
No se convierte a las páginas de códigos 924, 865, 1051, 1275

AIX

AIX

Página de códigos:

819
No se convierte a la página de códigos 865

Windows

Windows

Página de códigos:

865
No se convierte a las páginas de códigos 1051, 1275

Finés y sueco

Detalles de CCSID y conversión de CCSID para finés y sueco.

Tabla 645. CCSID nativos para finés y sueco en plataformas soportadas	
Plataforma	**CCSID nativos**
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 Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

278
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143
No se convierte a las páginas de códigos 865, 924, 1051, 1275

AIX

AIX

Página de códigos:

819
No se convierte a la página de códigos 865

850
No se convierte a la página de códigos 865

Windows

Windows

Página de códigos:

865
No se convierte a las páginas de códigos 1051, 1275

Italiano

Detalles de CCSID y conversión de CCSID para italiano.

Tabla 646. CCSID nativos para italiano en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	280, 924, 1144
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

280
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144
No se convierte a las páginas de códigos 924, 1051, 1275

Español

Detalles de CCSID y conversión de CCSID para español.

Tabla 647. CCSID nativos para español en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	284, 924, 1145
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

284
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145
No se convierte a las páginas de códigos 924, 1051, 1275

Inglés del Reino Unido /Gaélico

Detalles de CCSID y conversión de CCSID para inglés/gaélico del Reino Unido.

Tabla 648. CCSID nativos para inglés/gaélico del Reino Unido en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	285, 924, 1146
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

285
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146
No se convierte a las páginas de códigos 924, 1051, 1275

Francés

Detalles de CCSID y conversión de CCSID para francés.

Tabla 649. CCSID nativos para francés en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	297, 924, 1147
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

297
No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924
No se convierte a las páginas de códigos 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147
No se convierte a las páginas de códigos 924, 1051, 1275

Multi idioma

Detalles de CCSID y conversión de CCSID para multilingüe.

Tabla 650. CCSID nativos para la conversión multilingüe en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	500, 924, 1148
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

500
No se convierte a las páginas de códigos 858, 923

924
No se convierte a las páginas de códigos 437, 858, 1051, 1148, 1252, 1275, 5348

1148
No se convierte a las páginas de códigos 924, 1051, 1275

Portugués

Detalles de CCSID y conversión de CCSID para portugués.

Tabla 651. CCSID nativos para portugués en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i	37, 500, 924, 1140
z/OS IBM i	500, 924, 1140
AIX AIX	819, 923, 5348
Windows Windows	850, 858, 860, 1252, 5348
Linux Linux Solaris Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

37
No se convierte a las páginas de códigos 858, 923, 1275

500
No se convierte a las páginas de códigos 858, 923, 1275

924
No se convierte a las páginas de códigos 858, 860, 1051, 1140, 1252, 1275, 5348

1140
No se convierte a las páginas de códigos 860, 924, 1051, 1275

Windows

Windows

Página de códigos:

860
No se convierte a las páginas de códigos 1051, 1275

Islandés

Detalles de CCSID y conversión de CCSID para islandés.

Tabla 652. CCSID nativos para islandés en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	871, 924, 1149
AIX	819, 923, 5348
Windows	850, 858, 861, 1252, 5348
Linux Solaris	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

871
No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924
No se convierte a las páginas de códigos 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149
No se convierte a las páginas de códigos 924, 1051, 1275

Windows

Windows

Página de códigos:

861
No se convierte a las páginas de códigos 1051, 1275

Lenguas de Europa oriental

Detalles de CCSID y conversión de CCSID para idiomas de Europa oriental. Los idiomas típicos que utilizan estos CCSID son el albanés, el croata, el checo, el húngaro, el polaco, el rumano, el serbio, el eslovaco y el esloveno.

Tabla 653. CCSID nativos para idiomas de Europa oriental en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	870, 1153

Tabla 653. CCSID nativos para idiomas de Europa oriental en plataformas soportadas (continuación)	
Plataforma	**CCSID nativos**
Windows Windows	852, 1250, 5346, 9044
AIX AIX Linux Linux Solaris Solaris	912
Cliente de Apple de Europa oriental	1282
Cliente rumano de Apple	1285
Cliente de Apple croata	1284

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

870
No se convierte a las páginas de códigos 1284, 1285

1153
No se convierte a las páginas de códigos 1250, 1284, 1285

IBM i

IBM i

Página de códigos:

870
No se convierte a las páginas de códigos 1284, 1285, 5346, 9044

1153
No se convierte a las páginas de códigos 1282, 1284, 1285, 5346, 9044

Solaris Solaris, Linux

Solaris Linux

Página de códigos:

912
No se convierte a las páginas de códigos 1284, 1285

Windows

Windows

Página de códigos:

852
No se convierte a las páginas de códigos 1284, 1285

1250
No se convierte a las páginas de códigos 1284, 1285

9044
No se convierte a las páginas de códigos 912, 1282, 1284, 1285

Cirílico

Detalles de CCSID y conversión de CCSID para cirílico. Los idiomas típicos que utilizan estos CCSID son el bielorruso, el búlgaro, el macedonio, el ruso y el serbio.

Tabla 654. CCSID nativos para cirílico en plataformas soportadas

Plataforma	CCSID nativos
z/OS z/OS	1025
IBM i IBM i	880, 1025
Windows Windows	855, 866, 1131, 1251, 5347
Solaris Solaris	878, 915
AIX AIX Linux Linux	915
Cliente Apple	1283

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

880
No se convierte a las páginas de códigos 855, 866, 878, 1131, 5347

1025
No se convierte a las páginas de códigos 878, 5347

Windows

Windows

Página de códigos:

855
No se convierte a la página de códigos 1131

866
No se convierte a la página de códigos 1131

1131
No se convierte a las páginas de códigos 855, 866, 880, 1283

Estonio

Detalles de CCSID y conversión de CCSID para estonio.

Tabla 655. CCSID nativos para estonio en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	1122, 1157
Windows Windows	902, 922, 1257, 5353, 9449
AIX AIX Linux Linux Solaris Solaris	902, 922

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

1122
No se convierte a las páginas de códigos 902, 1157, 9449

1157
No se convierte a las páginas de códigos 922, 1122, 1257, 9449

IBM i

IBM i

Página de códigos:

1122
No se convierte a las páginas de códigos 902, 5353, 9449

1157
No se convierte a las páginas de códigos 922, 5353, 9449

Solaris, Linux

Solaris Linux

Página de códigos:

902
No se convierte a las páginas de códigos 922, 1122, 9449

922
No se convierte a las páginas de códigos 902, 1157, 9449

Windows

Windows

Página de códigos:

5353
No se convierte a la página de códigos 9449

9449
No se convierte a las páginas de códigos 902, 922, 1122, 1157, 1257, 5353

902
No se convierte a las páginas de códigos 922, 1122, 9449

Letón y lituano

Detalles de CCSID y conversión de CCSID para letón y lituano.

Tabla 656. CCSID nativos para letón y lituano en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	1112, 1156
Windows Windows	901, 921, 1257, 5353, 9449
AIX AIX Linux Linux Solaris Solaris	901, 921

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

1112
No se convierte a las páginas de códigos 901, 1156, 9449

1156
No se convierte a las páginas de códigos 901, 1156, 9449

IBM i

IBM i

Página de códigos:

1112
No se convierte a la página de códigos 5353

1153
No se convierte a las páginas de códigos 921, 5353, 9449

Solaris, Linux

Solaris Linux

Página de códigos:

902
No se convierte a las páginas de códigos 921, 1112, 1257, 9449

921
No se convierte a las páginas de códigos 901, 1156, 9449

Windows

Windows

Página de códigos:

901
No se convierte a las páginas de códigos 921, 1112, 1257, 9449

5355
No se convierte a la página de códigos 9449

9449
No se convierte a las páginas de códigos 901, 921, 1112, 1156, 1257

Ucraniano

Detalles de CCSID y conversión de CCSID para ucraniano.

Tabla 657. CCSID nativos para ucraniano en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	1123
Windows Windows	1124, 1125, 1251, 5347
AIX AIX Linux Linux Solaris Solaris	1124

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

1123
No se convierte a la página de códigos 5347

Windows

Windows

Página de códigos:

1125
No se convierte a la página de códigos 1123

Griego

Detalles de CCSID y conversión de CCSID para griego.

Tabla 658. CCSID nativos para griego en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	875
Windows Windows	869, 1253, 5349
AIX AIX Linux Linux NCR Solaris Solaris	813
Cliente Apple	1280
Cliente DOS	737

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos, los CCSID nativos de las otras plataformas con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

875
No se convierte a la página de códigos 5349

Windows

Windows

Página de códigos:

1253
No se convierte a la página de códigos 737

5349
No se convierte a la página de códigos 737

Turco

Detalles de CCSID y conversión de CCSID para turco.

Tabla 659. CCSID nativos para turco en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	1026
Windows Windows	857, 1254, 5350

Tabla 659. CCSID nativos para turco en plataformas soportadas (continuación)	
Plataforma	**CCSID nativos**
AIX AIX Linux Linux Solaris Solaris	920
Cliente Apple	1281

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

1026
No se convierte a la página de códigos 5350

Hebreo

Detalles de CCSID y conversión de CCSID para hebreo.

Tabla 660. CCSID nativos para hebreo en plataformas soportadas	
Plataforma	**CCSID nativos**
z/OS z/OS	424, 803, 4899, 12712
IBM i IBM i	424
AIX AIX	916, 9048
Windows Windows	1255, 5351
Linux Linux Solaris Solaris	916

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

424
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

803
No se convierte a las páginas de códigos 867, 4899, 5351, 9048, 12712

4899
No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

12712
No se convierte a las páginas de códigos 424, 803, 856, 916, 1255

IBM i

IBM i

Página de códigos:

424
No se convierte a las páginas de códigos 803, 867, 4899, 5351, 9048, 12712

La página de códigos 424 también se convierte a y desde CCSID 4952, que es una variante de 856.

AIX

AIX

Página de códigos:

916
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

9048
No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

Windows

Windows

Página de códigos:

1255
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

5351
No se convierte a la página de códigos 803

Árabe

Detalles de CCSID y conversión de CCSID para árabe

Tabla 661. CCSID nativos para árabe en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	420
AIX AIX	1046, 1089
	1089 (véase nota)
Windows Windows	720, 864, 1256, 5352
Linux Linux Solaris Solaris	1089

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

420
No se convierte a la página de códigos 5352

Solaris, Linux, Tru64

Solaris Linux

Página de códigos:

1089
No se convierte a la página de códigos 720

Windows

Windows

Página de códigos:

720
No se convierte a las páginas de códigos 1089, 5352

5352
No se convierte a la página de códigos 720

Farsi

Detalles de CCSID y conversión de CCSID para farsi.

Tabla 662. CCSID nativos para farsi en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	1097
AIX AIX Linux Linux Solaris Solaris Windows Windows	1098 (véase nota)

Nota: El CCSID nativo para estas plataformas no se ha estandarizado y puede cambiar.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Urdu

Detalles de CCSID y conversión de CCSID para Urdu.

Tabla 663. CCSID nativos para urdu en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	918
Windows	868
AIX Linux Solaris	1006

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

918
No se convierte a la página de códigos 1006

Tailandés

Detalles de CCSID y conversión de CCSID para tailandés.

Tabla 664. CCSID nativos para tailandés en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	838
AIX Linux Solaris Windows	874 (véase nota)

Nota: El CCSID nativo para estas plataformas no se ha estandarizado y puede cambiar.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Lao

Detalles de CCSID y conversión de CCSID para Lao.

Tabla 665. CCSID nativos para laosiano en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	1132
AIX Linux Solaris Windows	1133

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Vietnamita

Detalles de CCSID y conversión de CCSID para vietnamita.

Tabla 666. CCSID nativos para vietnamita en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i z/OS	1130
Windows	1258, 5354
AIX Linux Solaris	1129

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

IBM i

Página de códigos:

1130
No se convierte a las páginas de códigos 1129, 5354

Japonés latín SBCS

Detalles de CCSID y conversión de CCSID para SBCS latino japonés.

Tabla 667. CCSID nativos para SBCS latino japonés en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	1027
AIX AIX	932, 5050, 33722 (véase la nota 1)
Windows Windows	932, 943 (véase la nota 2)
Linux Linux Solaris Solaris	943, 5050

Nota:

1. AIX 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
2. Windows Windows NT utiliza la página de códigos 932, pero se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de IBM MQ dan soporte a este CCSID.

   En IBM MQ for Windows se utiliza el CCSID 932 para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

1027
No se convierte a las páginas de códigos 932, 942, 943, 954, 5050, 33722

IBM i

IBM i

Página de códigos:

1027
No se convierte a la página de códigos 932

AIX

AIX

Página de códigos:

932
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

33722
No se convierte a la página de códigos 1027

Linux

Linux

Página de códigos:

943
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

Solaris

Solaris

Página de códigos:

943
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

Japonés Katakana SBCS

Detalles de CCSID y conversión de CCSID para japonés Katakana SBCS.

Tabla 668. CCSID nativos para japonés Katakana SBCS en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	290
AIX AIX	932, 5050, 33722 (véase la nota 1)
Windows Windows	932, 943 (véase la nota 2)
Linux Linux Solaris Solaris	943, 5050

Nota:

1. AIX 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
2. Windows Windows NT utiliza la página de códigos 932, pero se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de IBM MQ dan soporte a este CCSID.

   En IBM MQ for Windows se utiliza el CCSID 932 para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.
3. Además de las conversiones anteriores, IBM MQ admite la conversión de CCSID 897 a CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 y 1252 en las plataformas siguientes:
   - AIX AIX
   - Linux Linux
   - Solaris Solaris

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

290
No se convierte a las páginas de códigos 932, 943, 954, 5050, 33722

IBM i

IBM i

Página de códigos:

290
No se convierte a la página de códigos 932

AIX

AIX

Página de códigos:

932
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

33722
No se convierte a las páginas de códigos 290, 897

Linux

Linux

Página de códigos:

943
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

Solaris

Solaris

Página de códigos:

943
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

Japonés Kanji/latín mixto

Detalles de CCSID y conversión de CCSID para japonés Kanji/Latin Mixed.

Tabla 669. CCSID nativos para japonés Kanji/Latin Mixed en plataformas soportadas	
Plataforma	**CCSID nativos**
IBM i IBM i z/OS z/OS	1399, 5035 (véase la nota 1)
AIX AIX	932, 5050, 33722 (véase la nota 2)
Windows Windows	932, 943 (véase la nota 4)
Linux Linux Solaris Solaris	943, 5050

Nota:

1. z/OS IBM i 5035 es un CCSID relacionado con la página de códigos 939
2. AIX 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
3. Windows Windows NT utiliza la página de códigos 932, pero se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de IBM MQ dan soporte a este CCSID.

   En IBM MQ for Windows se utiliza el CCSID 932 para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

1399
No se convierte a las páginas de códigos 954, 5035, 5050, 33722

5035
No se convierte a las páginas de códigos 954, 1399, 5050, 33722

IBM i

IBM i

Página de códigos:

1399
No se convierte a la página de códigos 5039

5035
No se convierte a la página de códigos 5039

Japonés Kanji/Katakana mixto

Detalles de CCSID y conversión de CCSID para japonés Kanji/Katakana mixto.

Tabla 670. CCSID nativos para japonés Kanji/Katakana mixto en plataformas soportadas	
Plataforma	**CCSID nativos**
z/OS z/OS	1390, 5026 (véase la nota 1)
IBM i IBM i	5026 (véase la nota 1)
AIX AIX	932, 5050, 33722 (véase la nota 2)
Windows Windows	932, 943 (véase la nota 4)
Linux Linux Solaris Solaris	943, 5050

Nota:

1. z/OS IBM i La modalidad de un solo byte de los CCSID 1390 y 5026 en EBCDIC contiene caracteres en minúsculas en diferentes ubicaciones para el diseño típico/invariante para el latín básico y se debe tener cuidado para asegurarse de que los datos no se pierdan cuando los datos del mensaje se están convirtiendo a otros CCSID. Además, el uso de estos CCSID como CCSID por omisión del gestor de colas puede provocar problemas al comunicarse con otros gestores de colas, por ejemplo, es posible que los nombres de canal que utilizan caracteres en minúsculas no se interpreten correctamente en el sistema remoto. 5026 es un CCSID relacionado con la página de códigos 930. CCSID 5026 es el CCSID notificado en IBM i cuando se selecciona la característica Katakana japonesa (DBCS).
2. AIX 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
3. Windows Windows NT utiliza la página de códigos 932, pero se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de IBM MQ dan soporte a este CCSID.

   En IBM MQ for Windows, el CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

1390
No convierte a páginas de códigos 954, 5026, 5050, 33722

No acepta caracteres en minúsculas.

5026
No se convierte a las páginas de códigos 954, 1390, 5050, 33722

IBM i

IBM i

Página de códigos:

5026
No se convierte a las páginas de códigos 1390, 5039

Coreano

Detalles de CCSID y conversión de CCSID para coreano.

Tabla 671. CCSID nativos para coreano en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	933, 1364
AIX AIX Linux Linux Solaris Solaris	970
Windows Windows	949, 1363

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

933
No se convierte a la página de códigos 970

1364
No se convierte a la página de códigos 970

Chino simplificado

Detalles de CCSID y conversión de CCSID para chino simplificado.

Tabla 672. CCSID nativos para chino simplificado en plataformas soportadas

Plataforma	CCSID nativos
z/OS z/OS	935, 1388
IBM i IBM i	935, 1388
AIX AIX	1383, 1386
Windows Windows	1381, 1386 (véase nota 2)
Linux Linux Solaris Solaris	1383

Nota:

1. Windows Windows utiliza la página de códigos 936, pero se representa mejor mediante el CCSID 1386. Sin embargo, no todas las plataformas de IBM MQ dan soporte a este CCSID.

En IBM MQ for Windows se utiliza el CCSID 1381 para representar la página de códigos 936, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 1386.

2. IBM MQ da soporte al estándar GB18030 en chino.

   Windows Solaris z/OS Linux En z/OS, Linux, Windowsy Solaris, el soporte de conversión se proporciona entre Unicode (UTF-8 y UTF-16) y CCSID 1388 (EBCDIC con extensiones GB18030), Unicode (UTF-8 y UTF-16) y CCSID 5488 (GB18030), y entre CCSID 1388 y CCSID 5488.

   Nota:

   IBM i En IBM i, el sistema operativo proporciona soporte para la conversión entre Unicode (UTF-8 y UTF-16) y CCSID 1388 (EBCDIC con extensiones GB18030).

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

935
No se convierte a la página de códigos 1383

1388
No se convierte a la página de códigos 1383

Chino tradicional

Detalles de CCSID y conversión de CCSID para chino tradicional.

Tabla 673. CCSID nativos para chino tradicional en plataformas soportadas

Plataforma	CCSID nativos
IBM i IBM i z/OS z/OS	937
Windows Windows	950
AIX AIX Linux Linux Solaris Solaris	950, 964

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

z/OS

Página de códigos:

937
No se convierte a la página de códigos 964

1388
No se convierte a la página de códigos 1383

Linux y Solaris

Solaris Linux

Página de códigos:

964
No se convierte a la página de códigos 938

z/OS Soporte de conversión de z/OS

Una lista de conversiones CCSID soportadas.

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS

CCSID	Convierte a y desde CCSIDS
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 13121, 13488, 16804, 17248, 17584, 28709
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1250, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
278	37, 256, 273, 277, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
281	1047
282	500, 1047, 1200, 1208, 13488, 17584
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
293	1200, 1208, 13488, 17584
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 1097, 1098, 1114-1115, 1126, 1140-1149, 1200, 1208, 1252, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210-5211, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-924, 1004, 1009-1021, 1023, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097, 1100-1107, 1112, 1114-1115, 1122, 1124-1126, 1129-1133, 1137, 1140-1149, 1200, 1208, 1250-1258, 1275, 1280-1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5142, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9061, 13121, 13488, 16804, 17584, 28709
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1051, 1088-1089, 1097, 1098, 1112, 1114, 1122-1123, 1126, 1130, 1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25617, 25619, 25664, 28709
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25479, 25617, 25619, 25664, 28709
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097, 1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1140, 1145-1146, 1148, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1148, 1149, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 9030, 12712, 13121, 13488, 16804, 17584, 28709
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869, 870, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5012, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 28709
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13488, 17584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584
1011	500, 1200, 1208, 13488, 17584
1012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 880, 897, 903, 912, 915-916, 920, 1009, 1026-1027, 1040-1043, 1051, 1088, 1112, 1122, 1131, 1200, 1208, 1251-1252, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5350, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1103	500
1104	500
1105	500
1106	500
1107	500
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200,1208,13488,17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1097-1098, 1112, 1122-1123, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1251, 1254-1255, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13488, 17584, 28709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1374	1200, 1208
1375	1200, 1208
1376	1200, 1208
1377	1200, 1208
1378	1200, 1208
1379	1200, 1208
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25473, 25617, 25619, 25664, 28709
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 25479, 25617, 25619, 25664, 28709
4933	837, 1200, 1208, 1380, 1385, 13488, 17584
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 17248, 25473, 25479, 25617, 25619, 28709
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097-1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 16804, 25473, 25479, 25617, 25619, 25664, 28709
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25479, 25617, 25619, 28709
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
8492	300-301, 941, 1351, 4396, 16684
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1153, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9238, 13121, 13488, 16804, 17248, 17584, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13488, 16804, 17248, 17584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13488, 17248, 17584, 25617, 25619, 25664, 28709
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 16684, 16804, 17248, 17584, 21427, 28709
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17584, 28709
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 21427, 28709
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 25479, 25617, 25619, 28709
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25617, 25619, 28709
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25619, 25664, 28709
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25617, 25664, 28709
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709

Tabla 674. Soporte de conversión CCSID de IBM MQ for z/OS (continuación)	
CCSID	**Convierte a y desde CCSIDS**
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

IBM i Soporte de conversión de IBM i

Puede encontrar una lista completa de los CCSID y las conversiones soportadas por IBM ien la publicación correspondiente de IBM i .

Las páginas de códigos soportadas se listan en Correlaciones de CCSID soportadas.

Soporte de conversión Unicode

Algunas plataformas dan soporte a la conversión de datos de usuario a o desde la codificación Unicode. Las dos formas de codificación Unicode soportadas son UTF-16 (CCSID 1200, 13488 y 17584) y UTF-8 (CCSID 1208). Debe utilizar los CCSID 1200 o 1208, ya que representan la versión Unicode más reciente soportada.

Los pares sustitutos UTF-16 (un par de caracteres UTF-16 de 2 bytes en el rango de X'D800'a X'DFFF' que representan un elemento de código Unicode por encima de U + FFFF) están soportados. Si un CCSID de destino no contiene una correlación para un elemento de código representado por un par sustituto UTF-16 , el par de caracteres se convierte en un único carácter de sustitución.

IBM MQda soporte a la combinación de secuencias de caracteres. Esto significa que, en algunos casos, un carácter precompuesto en el CCSID de origen se convertirá a una secuencia de caracteres de combinación en el CCSID de destino, o al revés.

Nota: IBM MQ no da soporte a los CCSID del gestor de colas UTF-16 , por lo que los datos de cabecera de mensaje no se pueden codificar en UTF-16.

Soporte de IBM MQ AIX para Unicode

AIX

En la conversión de IBM MQ for AIX a, y desde, los CCSID Unicode soportados (preferiblemente 1200 o 1208) están soportados para los CCSID no Unicode de la lista siguiente:

037
273, 278, 280, 284, 285, 297
423, 437
500
813, 819, 850, 852, 856, 857, 858, 860, 861, 865, 867, 869, 875, 878, 880
901, 902, 912, 915, 916, 920, 923, 924, 932, 933, 935, 937, 938, 939, 942, 943, 948, 949, 950, 954, 964, 970
1026, 1046, 1089
1129, 1130, 1131, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 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

Soporte de IBM MQ for Windows, Solaris, y Linux para Unicode

Windows Solaris Linux

En IBM MQ for Windows Solaris, IBM MQ for Solarisy IBM MQ para la conversión de Linux a, y desde, los CCSID Unicode soportados (preferiblemente 1200 o 1208) están soportados para los CCSID no Unicode de la lista siguiente:

037
277, 278, 280, 284, 285, 290, 297
300, 301
-420, 424, 437
500
813, 819, 833, 835, 836, 837, 838, 850, 852, 855, 856, 857, 858, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 874, 875, 878, 880, 891, 897
901, 902, 903, 904, 912, 913“5” en la página 1010, 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 931“1” en la página 1010, 932“2” en la página 1010, 933, 935, 937, 938“3” en la página 1010, 939, 941, 942, 943, 947, 948, 949, 950, 951, 954“4” en la página 1010, 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” en la página 1010
-9044, 9048, 9449
12712
13488
17584
33722“4” en la página 1010

Notas:

1. 931 utiliza 939 para la conversión.
2. 932 utiliza 942 para la conversión.
3. 938 utiliza 948 para la conversión.
4. 954 y 33722 utilizan 5050 para la conversión.
5. Sólo en Windows y Linux, y Solaris .

Soporte de IBM i para Unicode

IBM i

Para obtener detalles sobre el soporte de UNICODE, consulte la publicación correspondiente de IBM i relacionada con el sistema operativo.

Soporte de IBM MQ for z/OS para Unicode

z/OS

En la conversión de IBM MQ for z/OS a, y desde, los CCSID Unicode soportados (preferiblemente 1200 o 1208) están soportados para los CCSID no Unicode de la lista siguiente:

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

Estándares de codificación en plataformas de 64 bits

Utilice esta información para obtener información sobre los estándares de codificación en plataformas de 64 bits y los tipos de datos preferidos.

Tipos de datos preferidos

Estos tipos nunca cambian de tamaño y están disponibles tanto en plataformas IBM MQ de 32 bits como de 64 bits:

Tabla 675. Nombres y longitudes de tipos de datos

Nombre	Longitud
MQLONG	4 bytes
MQULONG	4 bytes
MQINT32	4 bytes
MQUINT32	4 bytes
MQINT64	8 bytes
MQUINT64	8 bytes

ULW Tipos de datos estándar en UNIX, Linuxy Windows

Conozca los tipos de datos estándar en aplicaciones UNIX y Linuxde 32 bits, UNIX y Linuxde 64 bits y Windows de 64 bits.

Aplicaciones UNIX y Linux de 32 bits

Linux UNIX

Esta sección se incluye para la comparación y se basa en Solaris. Las diferencias con otras plataformas UNIX se anotan:

Tabla 676. Nombres y longitudes de tipos de datos

Nombre	Longitud
car	1 byte
short	2 bytes
Int	4 bytes
largo	4 bytes
float	4 bytes
double	8 bytes

Tabla 676. Nombres y longitudes de tipos de datos (continuación)

Nombre	Longitud
doble largo	16 bytes
	Linux AIX Tenga en cuenta que en AIX y Linux PPC un doble largo es de 8 bytes.
puntero	4 bytes
ptrdiff_t	4 bytes
tamaño_t	4 bytes
tiempo_t	4 bytes
Clock_t	4 bytes
wchar_t	4 bytes
	AIX Tenga en cuenta que en AIX un wchar_t es de 2 bytes.

Aplicaciones UNIX y Linux de 64 bits

Linux UNIX

Esta sección se basa en Solaris. Las diferencias con otras plataformas UNIX se anotan:

Tabla 677. Nombres y longitudes de tipos de datos

Nombre	Longitud
car	1 byte
short	2 bytes
Int	4 bytes
largo	8 bytes
float	4 bytes
double	8 bytes
doble largo	16 bytes
	Linux AIX Tenga en cuenta que en AIX y Linux PPC un doble largo es de 8 bytes.
puntero	8 bytes
ptrdiff_t	8 bytes
tamaño_t	8 bytes
tiempo_t	8 bytes
Clock_t	8 bytes
	Tenga en cuenta que en la otra plataforma UNIX un clock_t es de 4 bytes.
wchar_t	4 bytes
	AIX Tenga en cuenta que en AIX un wchar_t es de 2 bytes.

Windows Aplicaciones de 64 bits

Windows

Tabla 678. Nombres y longitudes de tipos de datos

Nombre	Longitud
car	1 byte
short	2 bytes
Int	4 bytes
largo	4 bytes
float	4 bytes
double	8 bytes
doble largo	8 bytes
puntero	8 bytes Tenga en cuenta que todos los punteros son de 8 bytes.
ptrdiff_t	8 bytes
tamaño_t	8 bytes
tiempo_t	8 bytes
Clock_t	4 bytes
wchar_t	2 bytes
SEÑOR	2 bytes
DWORD	4 bytes
DESCRIPTOR de contexto	8 bytes
HFILE	4 bytes

Consideraciones sobre la codificación en Windows

Windows

HANDLE hf;

Utilizar

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

No utilizar

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

ya que esto produce un error.

size_t len fgets

Utilizar

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

No utilizar

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

imprf

Utilizar

```
printf("My struc pointer: %p", pMyStruc);
```

No utilizar

```
printf("My struc pointer: %x", pMyStruc);
```

Si necesita salida hexadecimal, tiene que imprimir los 4 bytes superiores e inferiores por separado.

char * ptr

Utilizar

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

No utilizar

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Utilizar

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

No utilizar

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

longitud

Utilizar

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

No utilizar

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

Sscanf

Utilizar

```
MQLONG SBCSprt;

sscanf(line, "%d", &SBCSprt);
```

No utilizar

```
MQLONG SBCSprt;

sscanf(line, "%1d", &SBCSprt);
```

`%ld` intenta poner un tipo de 8 bytes en un tipo de 4 bytes; solo utilice `%l` si está tratando con un tipo de datos `long` real. MQLONG, UINT32 y INT32 se definen como cuatro bytes, los mismos que un `int` en todas las plataformas IBM MQ :

IBM i Application Programming Reference (ILE/RPG)

Programación de aplicaciones para IBM i.

Utilice esta información como ayuda para desarrollar aplicaciones para IBM i.

- “Descripciones de tipos de datos en IBM i” en la página 1016
- “Llamadas de función en IBM i” en la página 1279
- “Atributos de objetos en IBM i” en la página 1401
- “Aplicaciones” en la página 1449
- “Códigos de retorno para IBM i (ILE RPG)” en la página 1462
- “Reglas para validar opciones MQI para IBM i (ILE RPG)” en la página 1464
- “Codificaciones de máquina en IBM i” en la página 1466
- “Opciones de informe y distintivos de mensaje en IBM i” en la página 1469

Desuso de la modalidad de compatibilidad para aplicaciones RPG y COBOL en IBM i

IBM i

Desde la IBM MQ 9.0, IBM MQ ya no proporciona soporte para aplicaciones RPG o COBOL que utilizan el enlace dinámico conocido como modalidad de compatibilidad. Esta modalidad de funcionamiento era necesaria para las aplicaciones escritas antes de MQSeries 5.1, y las versiones posteriores del producto proporcionaban un entorno de ejecución compatible para estas aplicaciones, aunque los libros de copias necesarios para compilarlas fueron eliminados en IBM WebSphere MQ 6.0. El enlace dinámico (modalidad de compatibilidad) se suministraba en los siguientes programas en la biblioteca QMQM, que se ha eliminado en IBM MQ 9.0:

- AMQVSTUB
- AMQZSTUB
- QMQM

- MQCLOSE
- MQCONN
- MQDISC
- MQGET
- MQINQ
- MQOPEN
- MQPUT
- MQPUT1
- MQSET

A partir de la IBM MQ 9.0, las aplicaciones que utilizan esta modalidad de compatibilidad de operación deben recompilarse para utilizar las llamadas MQ de enlace estático que proporcionan los programas de servicio LIBMQM y LIBMQM_R. Programas de ejemplo como AMQ3PUT4 y AMQ3GET4 muestran cómo utilizar este modelo de programación. Para obtener más información sobre la utilización de estas llamadas MQ, consulte IBM i Application Programming Reference (ILE/RPG).

Notas:

- Hay que recodificar las aplicaciones, que actualmente usan la interfaz CALL 'QMQM', para que usen el programa de servicio LIBMQM en su lugar.

  Los objetos de programa y programas de servicio de la lista anterior como, por ejemplo, QMQM, MQCONN, MQPUT, AMQVSTUB y AMQZSTUB, se han eliminado en IBM MQ 9.0 y las aplicaciones que se codificaron para usar el modo de compatibilidad dejan de funcionar.
- Si las aplicaciones están enlazadas al programa de servicio LIBMQM en IBM MQ 8.0, no es necesario que vuelva a compilar o enlazar estas aplicaciones en IBM MQ 9.0 o posterior.
- No es posible instalar más de una versión de IBM MQ para IBM i en la misma partición.

Si desea averiguar si el programa RPG o COBOL utiliza la modalidad de compatibilidad, utilice el mandato **DSPPGMREF** (Display Program References) para visualizar los programas externos llamados por el programa de aplicación. Si hay referencias a los programas listados en esta sección, el programa no se ejecutará en IBM MQ 9.0 o posterior. En el siguiente ejemplo de **DSPPGMREF**, la salida muestra objetos de programa que están en desuso MQCONN, MQOPEN, MQCLOSE:

```
Program . . . . . . . . . . . . . . . . . . . . :  MYAPPPGM
Library . . . . . . . . . . . . . . . . . . . . :    MYLIB
Text 'description'. . . . . . . . . . . . . . . :     ILE/COBOL SAMPLE PUT TO QUEUE (MQPUT)
Number of objects referenced  . . . . . . . . . :        5
Object  . . . . . . . . . . . . . . . . . . . . :    MQCONN
Library . . . . . . . . . . . . . . . . . . . . :      *LIBL
Object type . . . . . . . . . . . . . . . . . . :        *PGM
Object  . . . . . . . . . . . . . . . . . . . . :    MQOPEN
Library . . . . . . . . . . . . . . . . . . . . :      *LIBL
Object type . . . . . . . . . . . . . . . . . . :        *PGM
Object  . . . . . . . . . . . . . . . . . . . . :    MQCLOSE
Library . . . . . . . . . . . . . . . . . . . . :      *LIBL
Object type . . . . . . . . . . . . . . . . . . :        *PGM
```

Estos programas deben recompilarse utilizando el método Bound Procedural Call que se describe en Preparación de programas COBOL en IBM i.

Si intenta ejecutar un programa de aplicación en IBM MQ 9.0 o posterior que utiliza la modalidad de compatibilidad, el primer error que se ve con más frecuencia es un MCH3401 que intenta llamar al programa MQCONN o QMQM.

Tareas relacionadas

Desarrollo de aplicaciones

IBM i Descripciones de tipos de datos en IBM i

Esta colección de temas proporciona descripciones de los tipos de datos utilizados en la programación de IBM i .

Convenciones utilizadas en la descripción de tipos de datos

Para cada tipo de datos elemental, esta información proporciona una descripción de su uso, en un formato que es independiente del lenguaje de programación. Esto va seguido de declaraciones típicas en la versión ILE del lenguaje de programación RPG. Las definiciones de tipos de datos elementales se incluyen aquí para proporcionar coherencia. RPG utiliza especificaciones "D" donde los campos de trabajo pueden ser declarados usando los atributos que usted necesita. Sin embargo, puede hacerlo en las especificaciones de cálculo donde se utiliza el campo.

Para utilizar los tipos de datos elementales, cree:

- Un miembro /COPY que contiene todos los tipos de datos, o
- Estructura de datos externa (PF) que contiene todos los tipos de datos. A continuación, debe especificar los campos de trabajo con los atributos 'LIKE', el campo de tipo de datos adecuado.

Las ventajas de la segunda opción son que las definiciones se pueden utilizar como 'FIELD REFERENCE FILE' para otros objetos de IBM i . Si una definición de tipo de datos IBM MQ cambia, es relativamente sencillo volver a crear estos objetos.

Tipos de datos elementales

Todos los demás tipos de datos descritos en esta sección equivalen directamente a estos tipos de datos elementales o a agregados de estos tipos de datos elementales (matrices o estructuras).

Tabla 679. Tipos de datos elementales

Tipo de datos	Representación
MQBOOL	Entero con signo de 10 dígitos
MQBYTE	Campo alfanumérico de 1 byte
MQBYTE16	Campo alfanumérico de 16 bytes
MQBYTE24	Campo alfanumérico de 24 bytes
MQBYTE32	Campo alfanumérico de 32 bytes
MQBYTE64	Campo alfanumérico de 64 bytes
MQCHAR	Campo alfanumérico de 1 byte
MQCHAR4	Campo alfanumérico de 4 bytes
MQCHAR8	Campo alfanumérico de 8 bytes
MQCHAR12	Campo alfanumérico de 12 bytes
MQCHAR16	Campo alfanumérico de 16 bytes
MQCHAR20	Campo alfanumérico de 20 bytes
MQCHAR28	Campo alfanumérico de 28 bytes
MQCHAR32	Campo alfanumérico de 32 bytes
MQCHAR48	Campo alfanumérico de 48 bytes
MQCHAR64	Campo alfanumérico de 64 bytes
MQCHAR128	Campo alfanumérico de 128 bytes
MQCHAR256	Campo alfanumérico de 256 bytes
MQFLOAT32	Número de coma flotante de 4 bytes
MQFLOAT64	Número de coma flotante de 8 bytes

Tabla 679. Tipos de datos elementales (continuación)	
Tipo de datos	**Representación**
MQHCONFIG	Descriptor de contexto de configuración
MQHCONN	Entero con signo de 10 dígitos
MQHMSG	Manejador de mensajes que proporciona acceso a un mensaje
MQHOBJ	Entero con signo de 10 dígitos
MQINT8	Entero con signo de 8 bits
MQINT16	Entero con signo de 16 bits
MQINT32	Entero con signo de 32 bits
MQINT64	Entero con signo de 64 bits
MQLONG	Entero con signo de 32 bits
IDMQP	Identificador proceso
MQPTR	Puntero
MQTID	Identificador de hebra
MQUINT8	Entero sin signo de 8 bits
MQUINT16	Entero sin signo de 16 bits
MQUINT32	Entero de 32 bits sin signo
MQUINT64	Entero de 64 bits sin signo
MQULONG	Entero de 32 bits sin signo
PMQACH	Puntero a una estructura de datos de tipo MQACH
PMQAIR	Puntero a una estructura de datos de tipo MQAIR
PMQAXC	Puntero a una estructura de datos de tipo MQAXC
PMAXP	Puntero a una estructura de datos de tipo MAXP
PMQBMHO	Puntero a una estructura de datos de tipo MQBMHO
PMQBO	Puntero a una estructura de datos de tipo MQBO
PMQBOOL	Puntero a datos de tipo MQBOOL
PMQBYTE	Puntero a datos de tipo MQBYTE
PMQBYTEn	Puntero a datos de tipo MQBYTEn
PMQCBC	Puntero a una estructura de datos de tipo MQCBC
PMQCBD	Puntero a una estructura de datos de tipo MQCBD
PMQCHAR	Puntero a una estructura de datos de tipo MQCHAR
PMQCHARV	Puntero a una estructura de datos de tipo MQCHARV
PMQCHARn	Puntero a datos de tipo MQCHARn
PMQCIH	Puntero a una estructura de datos de tipo MQCIH

Tabla 679. Tipos de datos elementales (continuación)	
Tipo de datos	**Representación**
PMQCMHO	Puntero a una estructura de datos de tipo MQCMHO
PMQCNO	Puntero a una estructura de datos de tipo MQCNO
PMQCSP	Puntero a una estructura de datos de tipo MQCSP
PMQCTLO	Puntero a una estructura de datos de tipo MQCTLO
PMQDH	Puntero a una estructura de datos de tipo MQDH
PMQDHO	Puntero a una estructura de datos de tipo MQDHO
PMQDLH	Puntero a una estructura de datos de tipo MQDLH
PMQDMHO	Puntero a una estructura de datos de tipo MQDMHO
PMQDMPO	Puntero a una estructura de datos de tipo MQDMPO
PMQEPH	Puntero a una estructura de datos de tipo MQEPH
PMQFLOAT32	Puntero a datos de tipo MQFLOAT32
PMQFLOAT64	Puntero a datos de tipo MQFLOAT64
PMQFUNC	Puntero a una función
PMQGMO	Puntero a una estructura de datos de tipo MQGMO
PMQHCONFIG	Puntero a datos de tipo MQHCONFIG
PMQHCONN	Puntero a datos de tipo MQHCONN
PMQHMSG	Puntero a datos de tipo MQHMSG
PMQHOBJ	Puntero a datos de tipo MQHOBJ
PMQIIH	Puntero a una estructura de datos de tipo MQIIH
PMQIMPO	Puntero a una estructura de datos de tipo MQIMPO
PMQINT8	Puntero a datos de tipo MQINT8
PMQINT16	Puntero a datos de tipo MQINT16
PMQINT32	Puntero a datos de tipo MQINT32
PMQINT64	Puntero a datos de tipo MQINT64
PMQLONG	Puntero a datos de tipo MQLONG
PMQMD	Puntero a una estructura de datos de tipo MQMD
PMQMDE	Puntero a una estructura de datos de tipo MQMDE
PMQMD1	Puntero a una estructura de datos de tipo MQMD1
PMQMD2	Puntero a una estructura de datos de tipo MQMD2
PMQMHBO	Puntero a una estructura de datos de tipo MQMHBO
PMQOD	Puntero a una estructura de datos de tipo MQOD
PMQOR	Puntero a una estructura de datos de tipo MQOR

Tabla 679. Tipos de datos elementales (continuación)	
Tipo de datos	**Representación**
PMQPD	Puntero a una estructura de datos de tipo MQPD
PMQPID	Puntero a un identificador de proceso MQPID
PMQPMO	Puntero a una estructura de datos de tipo MQPMO
PMQPTR	Puntero a datos de tipo MQPTR
PMQRFH	Puntero a una estructura de datos de tipo MQRFH
PMQRFH2	Puntero a una estructura de datos de tipo MQRFH2
PMQRMH	Puntero a una estructura de datos de tipo MQRMH
PMQRR	Puntero a una estructura de datos de tipo MQRR
PMQSCO	Puntero a una estructura de datos de tipo MQSCO
PMQD	Puntero a una estructura de datos de tipo MQSD
PMQSMPO	Puntero a una estructura de datos de tipo MQSMPO
PMQSRO	Puntero a una estructura de datos de tipo MQSRO
PMQSTS	Puntero a una estructura de datos de tipo MQSTS
PMQTID	Puntero a un identificador de hebra MQTID
PMQTM	Puntero a una estructura de datos de tipo MQTM
PMQTMC2	Puntero a una estructura de datos de tipo MQTMC2
PMQUINT8	Puntero a datos de tipo MQUINT8
PMQUINT16	Puntero a datos de tipo MQUINT16
PMQUINT32	Puntero a datos de tipo MQUINT32
PMQUINT64	Puntero a datos de tipo MQUINT64
PMQULONG	Puntero a datos de tipo MQULONG
PMQVOID	Puntero
PMQWIH	Puntero a una estructura de datos de tipo MQWIH
PMQXQH	Puntero a una estructura de datos de tipo MQXQH

IBM i *MQBOOL en IBM i*

El tipo de datos MQBOOL representa un valor booleano. El valor 0 representa false. Cualquier otro valor representa true.

Un MQBOOL debe estar alineado como para el tipo de datos MQLONG.

IBM i *MQBYTE en IBM i*

El tipo de datos MQBYTE representa un único byte de datos.

Ninguna interpretación en particular se coloca en el byte-se trata como una serie de bits, y no como un número binario o carácter. No es necesaria ninguna alineación especial.

A veces se utiliza una matriz de MQBYTE para representar un área de almacenamiento principal con una naturaleza que no conoce el gestor de colas. Por ejemplo, el área puede contener datos de mensaje

de aplicación o una estructura. La alineación de los límites de esta zona debe ser compatible con la naturaleza de los datos contenidos en ella.

IBM i *MQBYTEn (Serie de n bytes) en IBM i*

Cada tipo de datos MQBYTEn representa una serie de *n* bytes.

Donde *n* puede tomar uno de los valores siguientes:

- 16, 24, 32 o 64.

Cada byte se describe mediante el tipo de datos MQBYTE. No es necesaria ninguna alineación especial.

Si los datos de la serie son más cortos que la longitud definida de la serie, los datos deben rellenarse con nulos para rellenar la serie.

Cuando el gestor de colas devuelve series de bytes a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con nulos la longitud definida de la serie.

Hay constantes disponibles que definen las longitudes de los campos de serie de bytes.

IBM i *MQCHAR (carácter) en IBM i*

El tipo de datos MQCHAR representa un único carácter.

El identificador de juego de caracteres codificado del carácter es el del gestor de colas (consulte el atributo **CodedCharSetId** en el tema CodedCharSetId). No es necesaria ninguna alineación especial.

Nota: Los datos de mensaje de aplicación especificados en las llamadas MQGET, MQPUT y MQPUT1 se describen mediante el tipo de datos MQBYTE, no el tipo de datos MQCHAR.

IBM i *MQCHARn (serie de n caracteres) en IBM i*

Cada tipo de datos MQCHARn representa una serie de *n* caracteres.

Donde *n* puede tomar uno de los valores siguientes:

- 4, 8, 12, 16, 20, 28, 32, 48, 64, 128 o 256

Cada carácter se describe mediante el tipo de datos MQCHAR. No es necesaria ninguna alineación especial.

Si los datos de la serie son más cortos que la longitud definida de la serie, los datos deben rellenarse con espacios en blanco para rellenar la serie. En algunos casos, se puede utilizar un carácter nulo para finalizar la serie de forma prematura, en lugar de rellenarla con espacios en blanco; el carácter nulo y los caracteres que le siguen se tratan como espacios en blanco, hasta la longitud definida de la serie. Los lugares donde se puede utilizar un nulo se identifican en las descripciones de llamada y tipo de datos.

Cuando el gestor de colas devuelve series de caracteres a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con blancos la longitud definida de la serie; el gestor de colas no utiliza el carácter nulo para delimitar la serie.

Hay constantes disponibles que definen las longitudes de los campos de serie de caracteres.

IBM i *MQFLOAT32 en IBM i*

El tipo de datos MQFLOAT32 es un número de coma flotante de 32 bits representado utilizando el formato de coma flotante IEEE estándar.

Un MQFLOAT32 debe estar alineado en un límite de 4 bytes.

IBM i *MQFLOAT64 en IBM i*

El tipo de datos MQFLOAT64 es un número de coma flotante de 64 bits representado utilizando el formato de coma flotante IEEE estándar.

Un MQFLOAT64 debe estar alineado en un límite de 8 bytes.

MQHCONFIG-descriptor de contexto de configuración

El tipo de datos MQHCONFIG representa un descriptor de contexto de configuración, es decir, el componente que se está configurando para un servicio instalable determinado. Un descriptor de contexto de configuración debe estar alineado en su límite natural.

Nota: Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

IBM i *MQHCONN (descriptor de conexión) en IBM i*

El tipo de datos MQHCONN representa un descriptor de conexión, es decir, la conexión con un gestor de colas determinado.

Un descriptor de conexión debe estar alineado en su límite natural.

Nota: Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

IBM i *MQHMSG (Descriptor de mensaje) en IBM i*

El tipo de datos **MQHMSG** representa un manejador de mensajes que proporciona acceso a un mensaje.

Un descriptor de mensaje debe estar alineado en un límite de 8 bytes.

Nota: Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

IBM i *MQHOBJ (descriptor de objeto) en IBM i*

El tipo de datos MQHOBJ representa un descriptor de contexto de objeto que proporciona acceso a un objeto.

Un descriptor de contexto de objeto debe estar alineado en su límite natural.

Nota: Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

IBM i *MQINT8 (entero con signo de 8 bits) en IBM i*

El tipo de datos MQINT8 es un entero con signo de 8 bits que puede tomar cualquier valor en el rango de -128 a +127, a menos que el contexto restrinja lo contrario.

IBM i *MQINT16 (entero con signo de 16 bits) en IBM i*

El tipo de datos MQINT16 es un entero con signo de 16 bits que puede tomar cualquier valor en el rango de -32 768 a +32 767, a menos que el contexto restrinja lo contrario.

Un MQINT16 debe estar alineado en un límite de 2 bytes.

IBM i *MQINT32 (entero de 32 bits) en IBM i*

El tipo de datos MQINT32 es un entero con signo de 32 bits.

Es equivalente a MQLONG.

IBM i *MQINT64 (entero de 64 bits) en IBM i*

El tipo de datos MQINT64 es un entero con signo de 64 bits que puede tomar cualquier valor en el rango de -9 223 372 036 854 775 808 a + 9 223 372 036 854 775 807, a menos que el contexto restrinja lo contrario.

Para COBOL, el rango válido está limitado a -999 999 999 999 999 999 a +999 999 999 999 999 999 999. Un MQINT64 debe estar alineado en un límite de 8 bytes.

IBM i *MQLONG (entero largo) en IBM i*

El tipo de datos MQLONG es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a + 2 147 483 647, a menos que esté restringido por el contexto, alineado en su límite natural.

MQPID-identificador de proceso
Identificador de proceso de IBM MQ .

Es el mismo identificador que se utiliza en los volcados IBM MQ trace y FFST , pero puede ser diferente del identificador de proceso del sistema operativo.

puntero MQPTR
El tipo de datos MQPTR es la dirección de los datos de cualquier tipo. Un puntero debe estar alineado en su límite natural; es un límite de 16 bytes en IBM i.

Algunos lenguajes de programación dan soporte a punteros con tipo; la MQI también los utiliza en unos pocos casos.

MQTID-identificador de hebra
El identificador de hebra de MQ .

Es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados FFST , pero puede ser diferente del identificador de hebra del sistema operativo.

IBM i *MQUINT8 (entero sin signo de 8 bits) en IBM i*
El tipo de datos MQUINT8 es un entero sin signo de 8 bits que puede tomar cualquier valor en el rango de 0 a +255, a menos que el contexto restrinja lo contrario.

MQUINT16 -entero sin signo de 16 bits
El tipo de datos MQUINT16 es un entero sin signo de 16 bits que puede tomar cualquier valor en el rango de 0 a +65 535, a menos que el contexto restrinja lo contrario.

Un MQUINT16 debe estar alineado en un límite de 2 bytes.

IBM i *MQUINT32 (entero sin signo de 32 bits) en IBM i*
El tipo de datos MQUINT32 es un entero sin signo de 32 bits. Es equivalente a MQULONG.

MQUINT64 -entero sin signo de 64 bits
El tipo de datos MQUINT64 es un entero sin signo de 64 bits que puede tomar cualquier valor en el rango de 0 a +18 446 744 073 709 551 615 a menos que el contexto restrinja lo contrario.

Para COBOL, el rango válido está limitado a 0 a +999 999 999 999 999 999. Un MQUINT64 debe estar alineado en un límite de 8 bytes.

MQULONG-entero sin signo de 32 bits
El tipo de datos MQULONG es un entero binario sin signo de 32 bits que puede tomar cualquier valor en el rango de 0 a + 4 294 967 294, a menos que el contexto restrinja lo contrario.

Un MQULONG debe estar alineado en un límite de 4 bytes.

PMQACH-puntero a una estructura de datos de tipo MQACH
Puntero a una estructura de datos de tipo MQACH.

PMQAIR-puntero a una estructura de datos de tipo MQAIR
Puntero a una estructura de datos de tipo MQAIR.

PMQAXC-puntero a una estructura de datos de tipo MQAXC
Puntero a una estructura de datos de tipo MQAXC.

PMQAXP-puntero a una estructura de datos de tipo MQAXP
Puntero a una estructura de datos de tipo MQAXP.

PMQBMHO-puntero a una estructura de datos de tipo MQBMHO
Puntero a una estructura de datos de tipo MQBMHO.

PMQBO-puntero a una estructura de datos de tipo MQBO
Puntero a una estructura de datos de tipo MQBO.

PMQBOOL-puntero a datos de tipo MQBOOL
Puntero a datos de tipo MQBOOL.

Puntero a datos de tipo MQBOOL.

PMQBYTE-puntero a un tipo de datos de MQBYTE
Puntero a un tipo de datos de MQBYTE.

PMQBYTEn-puntero a una estructura de datos de tipo MQBYTEn
Puntero a una estructura de datos de tipo MQBYTEn, donde n puede ser 8, 12, 16, 24, 32, 40, 48 o 128.

PMQCBC-puntero a una estructura de datos de tipo MQCBC
Puntero a una estructura de datos de tipo MQCBC.

PMQCBD-puntero a una estructura de datos de tipo MQCBD
Puntero a una estructura de datos de tipo MQCBD.

PMQCHAR-puntero a datos de tipo MQCHAR
Puntero a datos de tipo MQCHAR.

PMQCHARV-puntero a una estructura de datos de tipo MQCHARV
Puntero a una estructura de datos de tipo MQCHARV.

PMQCHARn-puntero a un tipo de datos de MQCHARn
Puntero a un tipo de datos de MQCHARn, donde n puede ser 4, 8, 12, 20, 28, 32, 64, 128, 256, 264.

PMQCIH-puntero a una estructura de datos de tipo MQCIH
Puntero a una estructura de datos de tipo MQCIH.

PMQCMHO-puntero a una estructura de datos de tipo MQCMHO
Puntero a una estructura de datos de tipo MQCMHO.

PMQCNO-puntero a una estructura de datos de tipo MQCNO
Puntero a una estructura de datos de tipo MQCNO.

PMQCSP-puntero a una estructura de datos de tipo MQCSP
Puntero a una estructura de datos de tipo MQCSP.

PMQCTLO-puntero a una estructura de datos de tipo MQCTLO
Puntero a una estructura de datos de tipo MQCTLO.

PMQDH-puntero a una estructura de datos de tipo MQDH
Puntero a una estructura de datos de tipo MQDH.

PMQDHO-puntero a una estructura de datos de tipo MQDHO
Puntero a una estructura de datos de tipo MQDHO.

PMQDLH-puntero a una estructura de datos de tipo MQDLH
Puntero a una estructura de datos de tipo MQDLH.

PMQDMHO-puntero a una estructura de datos de tipo MQDMHO
Puntero a una estructura de datos de tipo MQDMHO.

PMQDMPO-puntero a una estructura de datos de tipo MQDMPO
Puntero a una estructura de datos de tipo MQDMPO.

Puntero a una estructura de datos de tipo MQDMPO.

PMQEPH-puntero a una estructura de datos de tipo MQEPH
Puntero a una estructura de datos de tipo MQEPH.

PMQFLOAT32 -puntero a datos de tipo MQFLOAT32
Puntero a datos de tipo MQFLOAT32.

PMQFLOAT64 -puntero a datos de tipo MQFLOAT64
Puntero a datos de tipo MQFLOAT64.

PMQFUNC-puntero a una función
Puntero a una función.

PMQGMO-puntero a una estructura de datos de tipo MQGMO
Puntero a una estructura de datos de tipo MQGMO.

PMQHCONFIG-puntero a un tipo de datos MQHCONFIG
Puntero a un tipo de datos de MQHCONFIG.

PMQHCONN-puntero a un tipo de datos de MQHCONN
Puntero a un tipo de datos de MQHCONN.

PMQHMSG-puntero a un tipo de datos de MQHMSG
Puntero a un tipo de datos de MQHMSG.

PMQHOBJ-puntero a datos de tipo MQHOBJ
Puntero a datos de tipo MQSMPO.

PMQIIH-puntero a una estructura de datos de tipo MQIIH
Puntero a una estructura de datos de tipo MQIIH.

PMQIMPO-puntero a una estructura de datos de tipo MQIMPO
Puntero a una estructura de datos de tipo MQIMPO.

PMQINT8 -puntero a datos de tipo MQINT8
Puntero a datos de tipo MQINT8.

PMQINT16 -puntero a datos de tipo MQINT16
Puntero a datos de tipo MQINT16.

IBM i *PMQINT32 (Puntero a datos de tipo MQINT32) en IBM i*
El tipo de datos PMQINT32 es un puntero a datos de tipo MQINT32. Es equivalente a PMQLONG.

IBM i *PMQINT64 (Puntero a datos de tipo MQINT64) en IBM i*
El tipo de datos PMQINT64 es un puntero a datos de tipo MQINT64.

PMQLONG-puntero a datos de tipo MQLONG
Puntero a datos de tipo MQLONG.

PMQMD-puntero a estructura de tipo MQMD
Puntero a la estructura de tipo MQMD.

PMQMDE-puntero a una estructura de datos de tipo MQMDE
Puntero a una estructura de datos de tipo MQMDE.

PMQMDI-puntero a una estructura de datos de tipo MQMDI
Puntero a una estructura de datos de tipo MQMDI.

PMQMD2 : puntero a una estructura de datos de tipo MQMD2
Puntero a una estructura de datos de tipo MQMD2

PMQMHBO-puntero a una estructura de datos de tipo MQMHBO
Puntero a una estructura de datos de tipo MQMHBO.

PMQOD-puntero a una estructura de datos de tipo MQOD
Puntero a una estructura de datos de tipo MQOD.

PMQOR-puntero a una estructura de datos de tipo MQOR
Puntero a una estructura de datos de tipo MQOR.

PMQPD-puntero a una estructura de datos de tipo MQPD
Puntero a una estructura de datos de tipo MQPD.

PMQPID-puntero a un identificador de proceso
Puntero a un identificador de proceso.

PMQPMO-puntero a una estructura de datos de tipo MQPMO
Puntero a una estructura de datos de tipo MQPMO.

PMQPTR-puntero a datos de tipo MQPTR
Puntero a datos de tipo MQPTR.

PMQRFH-puntero a una estructura de datos de tipo MQRFH
Puntero a una estructura de datos de tipo MQRFH.

PMQRFH2 -puntero a una estructura de datos de tipo MQRFH2
Puntero a una estructura de datos de tipo MQRFH2.

.

PMQRMH-puntero a una estructura de datos de tipo MQRMH
Puntero a una estructura de datos de tipo MQRMH.

PMQRR-puntero a una estructura de datos de tipo MQRR
Puntero a una estructura de datos de tipo MQRR.

PMQSCO-puntero a una estructura de datos de tipo MQSCO
Puntero a una estructura de datos de tipo MQSCO.

.

PMQSD-puntero a una estructura de datos de tipo MQSD
Puntero a una estructura de datos de tipo MQSD.

PMQSMPO-puntero a una estructura de datos de tipo MQSMPO
Puntero a una estructura de datos de tipo MQSMPO.

PMQSRO-puntero a una estructura de datos de tipo MQSRO
Puntero a una estructura de datos de tipo MQSRO.

PMQSTS-puntero a una estructura de datos de tipo MQSTS
Puntero a una estructura de datos de tipo MQSTS.

PMQTID-puntero a una estructura de datos de tipo MQTID
Puntero a una estructura de datos de tipo MQTID.

PMQTM-puntero a una estructura de datos de tipo MQTM
Puntero a una estructura de datos de tipo MQTM.

PMQTMC2 -puntero a una estructura de datos de tipo MQTMC2
Puntero a una estructura de datos de tipo MQTMC2.

PMQUINT8 -puntero a datos de tipo MQUINT8
Puntero a datos de tipo MQUINT8.

PMQUINT16 -puntero a datos de tipo MQUINT16

Puntero a datos de tipo MQUINT16.

IBM i *PMQUINT32 (Puntero a datos de tipo MQUINT32) en IBM i*

El tipo de datos PMQUINT32 es un puntero a datos de tipo MQUINT32. Es equivalente a PMQULONG.

IBM i *PMQUINT64 (Puntero a datos de tipo MQUINT64) en IBM i*

El tipo de datos PMQUINT64 es un puntero a datos de tipo MQUINT64.

PMQULONG-puntero a datos de tipo MQULONG

Puntero a datos de tipo MQULONG.

PMQVOID-puntero

Un puntero.

PMQWIH-puntero a una estructura de datos de tipo MQWIH

Puntero a una estructura de datos de tipo MQWIH.

PMQXQH-puntero a una estructura de datos de tipo MQXQH

Puntero a una estructura de datos de tipo MQXQH.

Consideraciones sobre idiomas

Este tema contiene información para ayudarle a utilizar la MQI del lenguaje de programación RPG.

Algunas de estas consideraciones lingüísticas son:

- “Archivos de copia” en la página 1029
- “Llamadas” en la página 1032
- “parámetros de llamada” en la página 1032
- “Estructuras” en la página 1032
- “Constantes con nombre” en la página 1032
- “Procedimientos MQI” en la página 1032
- “Consideraciones sobre hebras” en la página 1033
- “Control de compromiso” en la página 1033
- “Codificación de las llamadas enlazadas” en la página 1033
- “convenios de anotación” en la página 1034

Archivos de copia

Se proporcionan varios archivos COPY para ayudar con la escritura de programas de aplicación RPG que utilizan la colocación de mensajes en colas. Hay tres conjuntos de archivos COPY:

- Los archivos COPY con nombres que terminan con la letra *G* se utilizan con programas que utilizan enlaces estáticos. Estos archivos se inicializan con las excepciones indicadas en "Estructuras" en la página 1032.
- Los archivos COPY con nombres que terminan con la letra *H* se utilizan con programas que utilizan enlaces estáticos, pero **no** se han inicializado.
- Los archivos COPY con nombres que terminan con la letra *R* se utilizan con programas que utilizan enlaces dinámicos. Estos archivos se inicializan con las excepciones indicadas en "Estructuras" en la página 1032.

Los archivos COPY residen en QRPGLESRC en la biblioteca QMQM.

Para cada conjunto de archivos COPY, hay dos archivos que contienen constantes con nombre y un archivo para cada una de las estructuras. Los archivos COPY se resumen en Tabla 680 en la página 1030.

Tabla 680. Archivos COPY de RPG

Nombre de archivo (enlace estático, inicializado, CMQ* G)	**Nombre de archivo (enlace estático, no inicializado, CMQ* H)**	**Nombre de archivo (enlace dinámico, inicializado, CMQ* R)**	**Contenido**
CMQBOG	CMQBOH	-	Estructura de opciones de inicio
CMQCDG	CMQCDH	CMQCDR	Estructura de definición de canal
CMQCFBFG	CMQCFBFH	-	Parámetro de filtro de bits PCF
CMQCFG	-	-	Constantes para PCF y sucesos
CMQCFBSG	CMQCFBSH	-	Serie de bytes PCF
CMQCFGRG	CMQCFGRH	-	parámetro de grupo PCF
CMQCFIFG	CMQCFIFH	-	Parámetro de filtro de enteros PCF
CMQCFHG	CMQCFHH	-	Cabecera PCF
CMQCFILG	CMQCFILH	-	Estructura de parámetros de lista de enteros PCF
CMQCFING	CMQCFINH	-	Estructura de parámetros enteros PCF
CMQCFSFG	CMQCFSFH	-	Parámetro de filtro de serie PCF
CMQCFSLG	CMQCFSLH	-	Estructura de parámetros de lista de series PCF
CMQCFSTG	CMQCFSTH	-	Estructura de parámetros de serie PCF
CMQCFXLG	CMQCFXLH	-	Nombre abreviado PCF para CFIL64
CMQCFXNG	CMQCFXNH	-	Nombre abreviado PCF para CFIN64
CMQCIHG	CMQCIHH	-	Estructura de cabecera de información de CICS
CMQCNOG	CMQCNOH	-	Estructura de opciones de conexión
CMQCSPG	CMQCSPH	-	Parámetros de seguridad
CMQCXPG	CMQCXPH	CMQCXPR	Estructura de parámetros de salida de canal
CMQDHG	CMQDHH	CMQDHR	Estructura de cabecera de distribución

Tabla 680. Archivos COPY de RPG (continuación)			
Nombre de archivo (enlace estático, inicializado, CMQ* G)	**Nombre de archivo (enlace estático, no inicializado, CMQ* H)**	**Nombre de archivo (enlace dinámico, inicializado, CMQ* R)**	**Contenido**
CMQDLHG	CMQDLHH	CMQDLHR	Estructura de cabecera de mensaje no entregado
CMQDXPG	CMQDXPH	CMQDXPR	Estructura de parámetros de salida de conversión de datos
CMQEPHG	CMQEPHH	-	Estructura de cabecera PCF incorporada
CMQG	-	CMQR	Constantes con nombre para MQI principal
CMQGMOG	CMQGMOH	CMQGMOR	Estructura de opciones de obtención de mensajes
CMQIIHG	CMQIIHH	CMQIIHR	Estructura de cabecera de información de IMS
CMQMDEG	CMQMDEH	CMQMDER	Estructura de extensión de descriptor de mensaje
CMQMDG	CMQMDH	CMQMDR	Estructura del descriptor de mensaje
CMQMD1G	CMQMD1H	CMQMD1R	Estructura de descriptor de mensaje versión 1
CMQMD2G	CMQMD2H	-	Estructura de descriptor de mensaje versión 2
CMQODG	CMQODH	CMQODR	Estructura de descriptor de objeto
CMQORG	CMQORH	CMQORR	Estructura de registro de objeto
CMQPMOG	CMQPMOH	CMQPMOR	Estructura de opciones de colocación de mensaje
CMQPSG	-	-	Constantes para publicación/suscripción
CMQRFHG	CMQRFHH	-	Reglas y estructura de cabecera de formato
CMQRFH2G	CMQRFH2H	-	Estructura de reglas y formateo de cabecera 2
CMQRMHG	CMQRMHH	CMQRMRRHH	Estructura de cabecera de mensaje de referencia
CMQRRG	CMQRRH	CMQRRR	Estructura de registro de respuesta
CMQTMCG	CMQTMCH	CMQTMCR	Estructura de mensaje desencadenante (formato de caracteres)
CMQTMC2G	CMQTMC2H	CMQTMC2R	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQTMG	CMQTMH	CMQTMR	Estructura de mensajes desencadenantes
CMQWIHG	CMQWIHH	-	Estructura de cabecera de información de trabajo

Tabla 680. Archivos COPY de RPG (continuación)			
Nombre de archivo (enlace estático, inicializado, CMQ* G)	**Nombre de archivo (enlace estático, no inicializado, CMQ* H)**	**Nombre de archivo (enlace dinámico, inicializado, CMQ* R)**	**Contenido**
CMQXG	-	CMQXR	Constantes con nombre para la salida de conversión de datos
CMQXQHG	CMQXQHH	CMQXQHR	Estructura de cabecera de cola de transmisión

Llamadas

Las llamadas se describen utilizando sus nombres individuales.

parámetros de llamada

Algunos parámetros pasados a la MQI pueden tener más de una función simultánea. Esto se debe a que el valor entero pasado a menudo se prueba en el valor de bits individuales dentro del campo, y no en su valor total. Esto le permite 'añadir' varias funciones juntas y pasarlas como un único parámetro.

Estructuras

Todas las estructuras IBM MQ se definen con valores iniciales para los campos, con las excepciones siguientes:

- Cualquier estructura con un sufijo de H.
- MQTMC
- MQTMC2

Estos valores iniciales se definen en la tabla correspondiente para cada estructura.

Las declaraciones de estructura no contienen sentencias DS . Esto permite a la aplicación declarar una sola estructura de datos o una estructura de datos de varias apariciones, codificando la sentencia DS y, a continuación, utilizando la sentencia /COPY para copiar en el resto de la declaración:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD        DS                          5
D/COPY CMQMDR
```

Constantes con nombre

Hay muchos valores enteros y de caracteres que proporcionan el intercambio de datos entre el programa de aplicación y el gestor de colas. Para facilitar un enfoque más legible y coherente para utilizar estos valores, se definen constantes con nombre para ellos. Puede utilizar estas constantes con nombre y no los valores que representan, ya que esto mejora la legibilidad del código fuente del programa.

Cuando el archivo COPY CMQG se incluye en un programa para definir las constantes, el compilador RPG emitirá muchos mensajes de gravedad cero para las constantes que no utiliza el programa; estos mensajes son benignos y se pueden ignorar de forma segura.

Procedimientos MQI

Al utilizar las llamadas enlazadas ILE, debe enlazar con los procedimientos MQI al crear el programa. Estos procedimientos se exportan desde los siguientes programas de servicio según corresponda:

QMQM/LIBMQM
Este programa de servicio contiene los enlaces de una sola hebra para la versión 5.1 y posteriores. Consulte la sección siguiente para obtener consideraciones especiales al escribir aplicaciones con hebras.

QMQM/LIBMQM_R
Este programa de servicio contiene los enlaces multihebra para la versión 5.1 y posteriores. Consulte la sección siguiente para obtener consideraciones especiales al escribir aplicaciones con hebras.

QMQM/LIBMQIC
Este programa de servicio es para enlazar aplicaciones cliente sin hebras.

QMQM/LIBMQIC_R
Este programa de servicio es para enlazar aplicaciones cliente con hebras.

Utilice el mandato `CRTPGM` para crear los programas. Por ejemplo, el mandato siguiente crea un programa de una sola hebra que utiliza las llamadas enlazadas ILE:

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

Consideraciones sobre hebras

El compilador RPG utilizado para IBM i forma parte de WebSphere Development Toolset y WebSphere Development Studio para IBM i y se conoce como compilador ILE RPG IV.

En general, los programas RPG no deben utilizar los programas de servicio multihebra. Las excepciones son programas RPG creados utilizando el compilador ILE RPG IV y que contienen la palabra clave `THREAD(*SERIALIZE)` en la especificación de control. Sin embargo, aunque estos programas sean seguros para hebras, se debe tener muy en cuenta el diseño general de la aplicación, ya que `THREAD(*SERIALIZE)` fuerza la serialización de los procedimientos RPG a nivel de módulo, y esto puede tener un efecto adverso en el rendimiento global.

Cuando los programas RPG se utilizan como salidas de conversión de datos, se deben convertir en hebras seguras y se deben volver a compilar utilizando el compilador ILE RPG de la versión 4.4 o superior, con `THREAD(*SERIALIZE)` especificado en la especificación de control.

Para obtener más información sobre las hebras, consulte *IBM i IBM MQ Development Studio: ILE RPG Reference*y *IBM i IBM MQ Development Studio: ILE RPG Programmer's Guide*.

Control de compromiso

Las funciones MQI de punto de sincronismo MQCMIT y MQBACK están disponibles para los programas ILE RPG que se ejecutan en modalidad normal; estas llamadas permiten al programa confirmar y restituir los cambios en los recursos de MQ .

Codificación de las llamadas enlazadas

Los procedimientos ILE de MQI se listan en Tabla 681 en la página 1033.

Tabla 681. Llamadas enlazadas ILE RPG soportadas por cada programa de servicio

Nombre de la llamada	LIBMQM y LIBMQM_R	LIBMQIC y LIBMQIC_R
MQBACK	Y	Y
MQBEGIN	Y	Y
MQCMIT	Y	Y
MQCLOSE	Y	Y
MQCONN	Y	Y
MQCONNX	Y	Y

Tabla 681. Llamadas enlazadas ILE RPG soportadas por cada programa de servicio (continuación)		
Nombre de la llamada	**LIBMQM y LIBMQM_R**	**LIBMQIC y LIBMQIC_R**
MQDISC	Y	Y
MQGET	Y	Y
MQINQ	Y	Y
MQOPEN	Y	Y
MQPUT	Y	Y
MQPUT1	Y	Y
MQSET	Y	Y
MQXCNVC	Y	Y

Para utilizar estos procedimientos, debe:

1. Defina los procedimientos externos en las especificaciones "D". Todos ellos están disponibles dentro del miembro de archivo COPY CMQG que contiene las constantes con nombre.
2. Utilice el código de operación CALLP para llamar al procedimiento junto con sus parámetros.

Por ejemplo, la llamada MQOPEN requiere la inclusión del código siguiente:

```
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*
```

Para llamar al procedimiento, después de inicializar los diversos parámetros, necesita el código siguiente:

```
...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+....8
    C                   CALLP     MQOPEN(HCONN : MQOD : OPTS : HOBJ :
    C                             CMPCOD : REASON)
```

Aquí, la estructura MQOD se define utilizando el miembro COPY CMQODG que lo divide en sus componentes.

convenios de anotación

Los últimos temas de esta sección muestran cómo:

- Se deben invocar las llamadas
- Los parámetros deben declararse
- Deben declararse varios tipos de datos

En varios casos, los parámetros son matrices o series de caracteres con un tamaño que no es fijo. Para estos, se utiliza una "n" en minúsculas para representar una constante numérica. Cuando la declaración para ese parámetro está codificada, "n" debe sustituirse por el valor numérico necesario.

IBM i MQAIR (registro de información de autenticación) en IBM i

La estructura MQAIR representa el registro de información de autenticación.

Visión general

Finalidad: la estructura MQAIR permite a una aplicación que se ejecuta como un cliente IBM MQ especificar información sobre un autenticador que se va a utilizar para la conexión de cliente. La estructura es un parámetro de entrada en la llamada MQCONNX.

Conjunto de caracteres y codificación: los datos de MQAIR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

- "Campos" en la página 1035
- "Valores iniciales" en la página 1037
- "Declaración RPG" en la página 1037

Campos

La estructura MQAIR contiene los campos siguientes; los campos se describen en **orden alfabético**:

AICN (entero con signo de 10 dígitos)

Es el nombre de host o la dirección de red de un host en el que se ejecuta el servidor LDAP. Esto puede ir seguido de un número de puerto opcional, entre paréntesis.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, la llamada falla con el código de razón RC2387.

El número de puerto predeterminado es 389.

Este es un campo de entrada. La longitud de este campo la proporciona LNAICN. El valor inicial de este campo es de caracteres en blanco.

AITYP (entero con signo de 10 dígitos)

Este es el tipo de información de autenticación contenida en el registro.

El valor debe ser:

AITLDP
Revocación de certificados utilizando el servidor LDAP.

Si el valor no es válido, la llamada falla con el código de razón RC2386.

Este es un campo de entrada. El valor inicial de este campo es AITLDP.

AIPW (entero con signo de 10 dígitos)

Esta es la contraseña necesaria para acceder al servidor CRL de LDAP.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el servidor LDAP no requiere una contraseña, u omite el nombre de usuario LDAP, *AIPW* debe ser nulo o estar en blanco. Si omite el nombre de usuario LDAP y *AIPW* no es nulo o está en blanco, la llamada falla con el código de razón RC2390.

Este es un campo de entrada. La longitud de este campo la proporciona LNLDPW. El valor inicial de este campo caracteres en blanco.

AILUL (entero con signo de 10 dígitos)

Es la longitud, en bytes, del nombre de usuario LDAP al que se dirige el campo *AILUP* o *AILUO* . El valor debe estar en el rango de cero a LNDISN. Si el valor no es válido, la llamada falla con el código de razón RC2389.

Si el servidor LDAP implicado no requiere un nombre de usuario, establezca este campo en cero.

Este es un campo de entrada. El valor inicial de este campo es 0.

AILUO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes del nombre de usuario LDAP desde el inicio de la estructura MQAIR.

El desplazamiento puede ser positivo o negativo. El campo se ignora si *LDAPUserNameLength* es cero.

Puede utilizar *LDAPUserNamePtr* o *LDAPUserNameOffset* para especificar el nombre de usuario LDAP, pero no ambos; consulte la descripción del campo *LDAPUserNamePtr* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

AILUP (entero con signo de 10 dígitos)

Es el nombre de usuario de LDAP.

Consta del nombre distinguido del usuario que está intentando acceder al servidor CRL de LDAP. Si el valor es más corto que la longitud especificada por *AILUL*, termine el valor con un carácter nulo, o rellena el valor con espacios en blanco hasta la longitud *AILUL*. El campo se ignora si *AILUL* es cero.

Puede proporcionar el nombre de usuario LDAP de una de estas dos maneras:

- Utilizando el campo de puntero *AILUP*

  En este caso, la aplicación puede declarar una serie que esté separada de la estructura MQAIR y establecer *AILUP* en la dirección de la serie.

  Considere la posibilidad de utilizar *AILUP* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

- Utilizando el campo de desplazamiento *AILUO*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga la estructura MQSCO seguida de la matriz de registros MQAIR seguida de las series de nombre de usuario LDAP y establecer *AILUO* en el desplazamiento de la serie de nombre adecuada desde el inicio de la estructura MQAIR. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Considere la posibilidad de utilizar *AILUO* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que podría no ser portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Cualquiera que sea la técnica elegida, utilice sólo una de *AILUP* y *AILUO* ; la llamada falla con el código de razón RC2388.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

AISID (entero con signo de 10 dígitos)

El valor debe ser:

AISIDV
Identificador del registro de información de autenticación.

Siempre es un campo de entrada. El valor inicial de este campo es AISIDV.

AIVER (entero con signo de 10 dígitos)

El valor debe ser:

AIVER1
Registro de información de autenticación Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

AIRVERC
Versión actual del registro de información de autenticación.

Siempre es un campo de entrada. El valor inicial de este campo es AIVER1.

Valores iniciales

Tabla 682. Valores iniciales de campos en MQAIR para MQAIR

Nombre de campo	Nombre de constante	Valor de constante
AISID	AISIDV	'AIR¬'
AIVER	AVERC	1
AITYP	AITLDP	1
AICN	Ninguna	Serie nula o espacios en blanco
AILUP	Ninguna	Puntero nulo o bytes nulos
AILUO	Ninguna	0
AILUL	Ninguna	0
AIPW	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (opciones de almacenamiento intermedio a manejador de mensajes) en IBM i

Estructura que define el almacenamiento intermedio para las opciones de manejo de mensajes.

Visión general

Finalidad: la estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Conjunto de caracteres y codificación: Los datos de MQBMHO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- "Campos" en la página 1038
- "Valores iniciales" en la página 1039
- "Declaración RPG" en la página 1039

Campos

La estructura MQBMHO contiene los campos siguientes; los campos se describen en **orden alfabético**:

BMSID (entero con signo de 10 dígitos)

Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo StrucId .

Es el identificador de estructura. El valor debe ser:

BMSIDV
Identificador de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es BMSIDV.

BMVER (entero con signo de 10 dígitos)

Estructura de almacenamiento intermedio a manejador de mensajes-Campo de versión.

Es el número de versión de la estructura. El valor debe ser:

BMVER1
Número de versión para la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

BMVERVC
Versión actual de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es BMVER1.

BMOPT (entero con signo de 10 dígitos)

Almacenamiento intermedio a estructura de descriptor de mensaje-Campo Opciones.

El valor puede ser:

BMDLPR
Las propiedades que se añaden al descriptor de contexto de mensaje se suprimen del almacenamiento intermedio. Si la llamada falla, no se suprimen las propiedades.

Opciones predeterminadas: Si no necesita la opción descrita, utilice la opción siguiente:

BMNONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es BMDLPR.

Valores iniciales

Tabla 683. Valores iniciales de los campos en MQBMHO		
Nombre de campo	**Nombre de constante**	**Valor de constante**
BMSID	BMSIDV	'BMHO'
BMVER	BMVER1	1
BMOPT	BMNONE	0

Declaración 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 (opciones de inicio) en IBM i

La estructura MQBO permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo.

Visión general

Finalidad: la estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Conjunto de caracteres y codificación: los datos de MQBO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

- "Campos" en la página 1039
- "Valores iniciales" en la página 1040
- "Declaración RPG" en la página 1040

Campos

La estructura MQBO contiene los campos siguientes; los campos se describen en **orden alfabético**:

BOOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQBEGIN.

El valor debe ser:

BONONA
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es BONONE.

BOSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

BOSIDV
Identificador de la estructura de opciones de inicio.

Siempre es un campo de entrada. El valor inicial de este campo es BOSIDV.

BOVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

BOVER1
Número de versión para la estructura de opciones de inicio.

La constante siguiente especifica el número de versión de la versión actual:

BOVERC
Versión actual de la estructura de opciones de inicio.

Siempre es un campo de entrada. El valor inicial de este campo es BOVER1.

Valores iniciales

Tabla 684. Valores iniciales de campos en MQBO

Nombre de campo	Nombre de constante	Valor de constante
BOSID	BOSIDV	'BO¬¬'
BOVER	BOVER1	1
BOOPT	BONONA	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (contexto de devolución de llamada) en IBM i

Estructura que describe la rutina de devolución de llamada.

Visión general

Finalidad
La estructura MQCBC se utiliza para especificar información de contexto que se pasa a una función de devolución de llamada.

La estructura es un parámetro de entrada/salida en la llamada a una rutina de consumidor de mensajes.

Versión
La versión actual de MQCBC es CBCV2.

Juego de caracteres y codificación
Los datos de MQCBC están en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por ENNAT. Sin embargo,

si la aplicación se ejecuta como un cliente IBM MQ , la estructura está en el juego de caracteres y la codificación del cliente.

- “Campos” en la página 1041
- “Valores iniciales” en la página 1046
- “Declaración RPG” en la página 1047

Campos

La estructura MQCBC contiene los campos siguientes; los campos se describen en orden alfabético:

CBCBUFFLEN (entero con signo de 10 dígitos)

El almacenamiento intermedio puede ser mayor que el valor de longitud MaxMsgdefinido para el consumidor y el valor de ReturnedLength en el MQGMO.

Estructura de contexto de devolución de llamada-campo BufferLength .

Es la longitud en bytes del almacenamiento intermedio de mensajes que se ha pasado a esta función.

La longitud real del mensaje se proporciona en el campo DataLength .

La aplicación puede utilizar todo el almacenamiento intermedio para sus propios fines durante la duración de la función de devolución de llamada.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de excepciones.

CBCCALLBA (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo CallbackArea .

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo CBDCALLBA en la estructura MQCBD, que es un parámetro de la llamada MQCB utilizada para definir la función de devolución de llamada.

Los cambios en *CBCCALLBA* se conservan en las invocaciones de la función de devolución de llamada para un *CBCHOBJ*. Este campo no se comparte con las funciones de devolución de llamada para otros manejadores.

Es un campo de entrada/salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBCCALLT (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo CallType .

Campo que contiene información sobre por qué se ha llamado a esta función. Se definen los siguientes tipos de llamada.

Tipos de llamada de entrega de mensajes: estos tipos de llamada contienen información sobre un mensaje. Los parámetros **CBCLEN** y **CBCBUFFLEN** son válidos para estos tipos de llamada.

CBCTMR
La función de consumidor de mensajes se ha invocado con un mensaje que se ha eliminado de forma destructiva del descriptor de contexto de objeto.

Si el valor de *CBCCC* es CCWARN, el valor del campo *Reason* es RC2079 o uno de los códigos que indican un problema de conversión de datos.

CCTMN
La función de consumidor de mensajes se ha invocado con un mensaje que todavía no se ha eliminado de forma destructiva del descriptor de objeto. El mensaje se puede eliminar de forma destructiva del descriptor de objeto utilizando *MsgToken*.

Es posible que el mensaje no se haya eliminado porque:

- Las opciones MQGMO solicitaron una operación de examen, GMBR*
- El mensaje es mayor que el almacenamiento intermedio disponible y las opciones MQGMO no especifican `gmatm`

Si el valor de *CBCCC* es CCWARN, el valor del campo *Reason* es `RC2080` o uno de los códigos que indican un problema de conversión de datos.

Tipos de llamada de control de devolución de llamada: estos tipos de llamada contienen información sobre el control de la devolución de llamada y no contienen detalles sobre un mensaje. Estos tipos de llamada se solicitan utilizando CBDOPT en la estructura MQCBD.

Los parámetros **CBCLEN** y **CBCBUFFLEN** no son válidos para estos tipos de llamada.

CBCTRC
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración inicial.

La función de devolución de llamada se invoca inmediatamente después de que se registre la devolución de llamada, es decir, al volver de una llamada MQCB utilizando un valor para el campo *Operation* de CBREG.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, es la primera invocación de la función de devolución de llamada.

El valor del campo *CBCREA* es RCNONE.

CBCTSC
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración cuando se inicia, por ejemplo, restableciendo los recursos que se han limpiado cuando se detuvo anteriormente.

La función de devolución de llamada se invoca cuando se inicia la conexión utilizando CTLSR o CTLSW.

Si una función de devolución de llamada se registra dentro de otra función de devolución de llamada, este tipo de llamada se invoca cuando se devuelve la devolución de llamada.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *CBCREA* es RCNONE.

CCTTC
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna limpieza cuando se detiene durante un tiempo, por ejemplo, limpiando recursos adicionales que se han adquirido durante el consumo de mensajes.

La función de devolución de llamada se invoca cuando se emite una llamada MQCTL utilizando un valor para el campo *Operation* de CTLSP.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *CBCREA* se establece para indicar la razón de la detención.

CBCTDC
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice la limpieza final al final del proceso de consumo. La función de devolución de llamada se invoca cuando:

- La función de devolución de llamada se anula del registro utilizando una llamada MQCB con BCUNR.
- La cola está cerrada, lo que provoca un anulación de registro implícito. En este caso, se pasa la función de devolución de llamada HOUNUH como descriptor de contexto de objeto.

- La llamada MQDISC se completa-causando un cierre implícito y, por lo tanto, un anulación del registro. En este caso, la conexión no se desconecta inmediatamente y cualquier transacción en curso todavía no se confirma.

Si alguna de estas acciones se realiza dentro de la propia función de devolución de llamada, la acción se invoca una vez que se devuelve la devolución de llamada.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, esta es la última invocación de la función de devolución de llamada.

El valor del campo *CBCREA* se establece para indicar la razón de la detención.

CCTEC
Función de manejador de sucesos

La función de manejador de sucesos se ha invocado sin un mensaje cuando:

- Se emite una llamada MQCTL con un valor para el campo *Operation* de CTLSP, o
- El gestor de colas o la conexión se detiene o inmoviliza.

Esta llamada se puede utilizar para realizar la acción adecuada para todas las funciones de devolución de llamada.

- **Función de consumidor de mensajes**

  La función de consumidor de mensajes se ha invocado sin un mensaje cuando se ha detectado un error (*CBCCC* = CCFAIL) que es específico del descriptor de objeto; por ejemplo, *CBCREA* code = RC2016 .

El valor del campo *CBCREA* se establece para indicar la razón de la llamada.

Este es un campo de entrada. CBCTMR y CMCTMN sólo son aplicables a las funciones de consumidor de mensajes.

CBCCC (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo CompCode .

Este es el código de terminación. Indica si se han producido problemas al consumir el mensaje; es uno de los siguientes:

CCOK
Realización satisfactoria

CCWARN
Aviso (terminación parcial)

CCFAIL
Llamada fallida

Este es un campo de entrada. El valor inicial de este campo es CCOK.

CBCCONNAREA (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo ConnectionArea .

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo ConnectionArea de la estructura MQCTLO, que es un parámetro de la llamada MQCTL utilizada para controlar la función de devolución de llamada.

Los cambios realizados en este campo por las funciones de devolución de llamada se conservan en las invocaciones de la función de devolución de llamada. Esta área se puede utilizar para pasar información que deben compartir todas las funciones de devolución de llamada. A diferencia de *CallbackArea*, esta área es común en todas las devoluciones de llamada para un descriptor de conexión.

Es un campo de entrada y salida. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBCLEN (entero con signo de 10 dígitos)

Es la longitud en bytes de los datos de aplicación del mensaje. Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

El campo CBCLEN contiene la longitud del mensaje, pero no necesariamente la longitud de los datos de mensaje pasados al consumidor. Puede ser que el mensaje se haya truncado. Utilice el campo GMRL en MQGMO para determinar cuántos datos se han pasado al consumidor.

Si el código de razón indica que el mensaje se ha truncado, puede utilizar el campo CBCLEN para determinar el tamaño del mensaje real. Esto le permite determinar el tamaño del almacenamiento intermedio necesario para acomodar los datos del mensaje y, a continuación, emitir una llamada MQCB para actualizar CBDMML en MQCBD con un valor adecuado.

Si se especifica la opción GMCONV, el mensaje convertido podría ser mayor que el valor devuelto para DataLength. En tales casos, es probable que la aplicación tenga que emitir una llamada MQCB para actualizar CBDMML en MQCBD para que sea mayor que el valor devuelto por el gestor de colas para DataLength.

Para evitar problemas de truncamiento de mensajes, especifique MaxMsgLength como CBDFM. Esto hace que el gestor de colas asigne un almacenamiento intermedio para la longitud completa del mensaje después de la conversión de datos. Sin embargo, tenga en cuenta que incluso si se especifica esta opción, todavía es posible que no haya suficiente almacenamiento disponible para procesar correctamente la solicitud. Las aplicaciones siempre deben comprobar el código de razón devuelto. Por ejemplo, si no es posible asignar almacenamiento suficiente para convertir el mensaje, los mensajes se devuelven a la aplicación sin convertir.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

CBCFLG (entero con signo de 10 dígitos)

Distintivos que contienen información sobre este consumidor.

Se define la opción siguiente:

CCBBE

Este distintivo se puede devolver si una llamada MQCLOSE anterior que utilizaba la opción COQSC ha fallado con un código de razón de RC2458.

Este código indica que se está devolviendo el último mensaje de lectura anticipada y que el almacenamiento intermedio está ahora vacío. Si la aplicación emite otra llamada MQCLOSE utilizando la opción COQSC, se ejecuta correctamente.

Tenga en cuenta que no se garantiza que a una aplicación se le asigne un mensaje con este distintivo establecido, ya que es posible que todavía haya mensajes en el almacenamiento intermedio de lectura anticipada que no coincidan con los criterios de selección actuales. En este caso, la función de consumidor se invoca con el código de razón RC2019 .

Si el almacenamiento intermedio de lectura anticipada está vacío, el consumidor se invoca con el distintivo CBCFBE y el código de razón RC2518.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

CBCHOBJ (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo CBCHOBJ.

Para una llamada a un consumidor de mensajes, este es el descriptor de contexto del objeto relacionado con el consumidor de mensajes.

Para un manejador de sucesos, este valor es HONONE

La aplicación puede utilizar este descriptor de contexto y la señal de mensaje en el bloque Obtener opciones de mensaje para obtener el mensaje si no se ha eliminado un mensaje de la cola.

Siempre es un campo de entrada. El valor inicial de este campo es HOUNUH

CBCRCD (entero con signo de 10 dígitos)

CBCRCD indica cuánto tiempo espera el gestor de colas antes de intentar volver a conectarse. El campo puede ser modificado por un manejador de sucesos para cambiar el retardo o detener la reconexión por completo.

Utilice el campo **CBCRCD** sólo si el valor del campo **Reason** en el Contexto de devolución de llamada es RC2545.

En la entrada al manejador de sucesos, el valor de **CBCRCD** es el número de milisegundos que el gestor de colas va a esperar antes de realizar un intento de reconexión. La Tabla 685 en la página 1045 lista los valores que puede establecer para modificar el comportamiento del gestor de colas en el retorno del manejador de sucesos.

*Tabla 685. **CBCRCD** los valores*

Valor	Descripción
`-1`	No realice más intentos de reconexión. Se devuelve un error a la aplicación.
`0`	Intente volver a conectarse inmediatamente.
`>0`	Espere este número de milisegundos antes de volver a intentar la conexión.

CBCREA (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo de razón.

Este es el código de razón que califica el *CBCCC*

Este es un campo de entrada. El valor inicial de este campo es RCNONE.

CBCSTATE (entero con signo de 10 dígitos)

Indicación del estado del consumidor actual. Este campo es de mayor valor para una aplicación cuando se pasa un código de razón distinto de cero a la función de consumidor.

Puede utilizar este campo para simplificar la programación de aplicaciones porque no es necesario codificar el comportamiento para cada código de razón.

Este es un campo de entrada. El valor inicial de este campo es CSNONE

Tabla 686. Valores CBCSTATE y acciones resultantes

Estado	Acción del gestor de colas	Valor de constante
CSNONE Este código de razón representa una llamada normal sin información de razón adicional	No; esta es la operación normal.	`0`
CSSUST Estos códigos de razón representan condiciones temporales.	Se llama a la rutina de devolución de llamada para informar de la condición y, a continuación, se suspende. Después de un periodo, el sistema puede volver a intentar la operación, lo que puede llevar a que se vuelva a generar la misma condición.	`1`

Tabla 686. Valores CBCSTATE y acciones resultantes (continuación)		
Estado	**Acción del gestor de colas**	**Valor de constante**
CSSUSU Estos códigos de razón representan condiciones en las que la devolución de llamada debe actuar para resolver la condición.	El consumidor se suspende y se llama a la rutina de devolución de llamada para informar de la condición. La rutina de devolución de llamada debe resolver la condición si es posible y RESUME o cerrar la conexión.	2
CSSUS Estos códigos de razón representan anomalías que impiden más devoluciones de llamada de mensaje.	El gestor de colas suspende automáticamente la función de devolución de llamada. Si se reanuda la función de devolución de llamada, es probable que vuelva a recibir el mismo código de razón.	3
CSSTOP Estos códigos de razón representan el final del consumo de mensajes.	Se entrega al manejador de excepciones y a las devoluciones de llamada que han especificado CBDTC. No se pueden consumir más mensajes.	4

CBCSID (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-campo StrucId .

Este es el identificador de estructura; el valor debe ser:

CCIC
Identificador de la estructura de contexto de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es CBCSI.

CBCVER (entero con signo de 10 dígitos)

Estructura de contexto de devolución de llamada-Campo Versión.

Este es el número de versión de la estructura; el valor debe ser:

CBCV1
Estructura de contexto de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

CCCV
Versión actual de la estructura de contexto de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es CBCV1.

Valores iniciales

Tabla 687. Valores iniciales de campos en MQCBC		
Nombre de campo	**Nombre de constante**	**Valor de constante**
CBCSID	CCIC	`'CBC¬'`
CBCVER	CBCV1	`1`
CBCCALLT	Ninguna	`0`
CBCHOBJ	HOUNUH	`-1`
CBCCALLBA	Ninguna	Puntero nulo o bytes nulos

Tabla 687. Valores iniciales de campos en MQCBC (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
CBCCONNAREA	Ninguna	Puntero nulo o bytes nulos
CBCCC	CCOK	0
CBCREA	RCNONE	0
CBCSTATE	CSNONE	0
CBCLEN	Ninguna	0
CBCBUFFLEN	Ninguna	0
CBCFLG	Ninguna	0
CBCRCD	ninguno	0

Nota:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Descriptor de devolución de llamada) en IBM i

Estructura que especifica la función de devolución de llamada.

Visión general

Finalidad: la estructura MQCBD se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas.

La estructura es un parámetro de entrada en la llamada MQCB.

Versión: La versión actual de MQCBD es CBDV1.

Conjunto de caracteres y codificación: los datos de MQCBD deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas **CodedCharSetId** y ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

- "Campos" en la página 1048
- "Valores iniciales" en la página 1052
- "Declaración RPG" en la página 1052

Campos

La estructura MQCBD contiene los campos siguientes; los campos se describen en **orden alfabético**:

CBDCALLBA (entero con signo de 10 dígitos)

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo CBCCALLBA en la estructura MQCBD, que es un parámetro en la declaración de función de devolución de llamada.

El valor sólo se utiliza en un *Operation* que tenga un valor CBREG, sin ninguna devolución de llamada definida actualmente, no sustituye a una definición anterior.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBDCALLBF (entero con signo de 10 dígitos)

La función de devolución de llamada se invoca como una llamada de función.

Utilice este campo para especificar un puntero a la función de devolución de llamada.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón RC2486 .

Si no se establece *CallbackName* ni *CallbackFunction* , la llamada falla con el código de razón RC2486.

Esta opción no está soportada en los entornos siguientes:

- CICS en z/OS
- Lenguajes de programación y compiladores que no dan soporte a referencias de puntero de función

En tales situaciones, la llamada falla con el código de razón RC2486.

Este es un campo de entrada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBDCALLBN (entero con signo de 10 dígitos)

La función de devolución de llamada se invoca como un programa enlazado dinámicamente.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón RC2486 .

Si *CallbackName* o *CallbackFunction* no es verdadero, la llamada falla con el código de razón RC2486.

El módulo se carga cuando se registra la primera rutina de devolución de llamada que se va a utilizar y se descarga cuando se anula el registro de la última rutina de devolución de llamada que se va a utilizar.

Excepto cuando se indica en el texto siguiente, el nombre se alinea a la izquierda dentro del campo, sin blancos intercalados; el propio nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

IBMi

El nombre de devolución de llamada puede tener uno de los formatos siguientes:

- Programa de biblioteca "/"
- Library "/" ServiceProgram "("FunctionName")"

Por ejemplo, `MyLibrary/MyProgram(MyFunction)`.

El nombre de biblioteca puede ser *LIBL. Los nombres de biblioteca y de programa están limitados a un máximo de 10 caracteres.

UNIX

El nombre de devolución de llamada es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[path]library(function)
```

Si no se especifica la vía de acceso, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

Windows

El nombre de devolución de llamada es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio y una unidad:

```
[d:][path]library(function)
```

Si la unidad y la vía de acceso no se especifican, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD.

El nombre está limitado a un máximo de 8 caracteres.

z/OS CICS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro PROGRAM de la macro de mandato EXEC CICS LINK.

El nombre está limitado a un máximo de 8 caracteres.

El programa puede definirse como remoto utilizando la opción REMOTESYTEM de la definición PROGRAM instalada o mediante el programa de direccionamiento dinámico.

La región CICS remota debe estar conectada a IBM MQ si el programa va a utilizar llamadas de API de IBM MQ . Sin embargo, tenga en cuenta que el campo CBCHOBJ de la estructura MQCBC no es válido en un sistema remoto.

Si se produce una anomalía al intentar cargar *CallbackName*, se devuelve uno de los siguientes códigos de error a la aplicación:

- RC2495
- RC2496
- RC2497

También se graba un mensaje en el registro de errores que contiene el nombre del módulo para el que se ha intentado la carga y el código de razón de anomalía del sistema operativo.

Este es un campo de entrada. El valor inicial de este campo es una serie nula o espacios en blanco.

CBDCALLBT (entero con signo de 10 dígitos)

Es el tipo de la función de devolución de llamada. El valor debe ser uno de los siguientes:

CBTMC
Define esta devolución de llamada como una función de consumidor de mensajes.

Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto y se inicia la conexión.

CTEH
Define esta devolución de llamada como la rutina de suceso asíncrona; no se controla que consuma mensajes para un descriptor de contexto.

Hobj no es necesario en la llamada MQCB que define el manejador de sucesos y se ignora si se especifica.

Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de consumidor de mensajes. La función de consumidor se invoca sin un mensaje cuando se produce un suceso, por ejemplo, un gestor de colas o una conexión deteniéndose o desactivándose temporalmente. No se llama para condiciones específicas de un único consumidor de mensajes, por ejemplo, RC2016.

Los sucesos se entregan a la aplicación, independientemente de si la conexión se ha iniciado o detenido, excepto en los entornos siguientes:

- Entorno CICS en z/OS
- aplicaciones sin hebras

Si el llamante no pasa uno de estos valores, la llamada falla con un código de razón de RC2483

Siempre es un campo de entrada. El valor inicial de este campo es CBTMC.

CBDMML (entero con signo de 10 dígitos)

Es la longitud en bytes del mensaje más largo que se puede leer desde el descriptor de contexto y se puede asignar a la rutina de devolución de llamada. Si un mensaje tiene una longitud mayor, la rutina de devolución de llamada recibe *MaxMsgLength* bytes del mensaje y el código de razón:

- RC2080 o
- RC2079 si ha especificado GMATM.

La longitud real del mensaje se proporciona en el campo "CBCLEN (entero con signo de 10 dígitos)" en la página 1044 de la estructura MQCBC.

Se define el siguiente valor especial:

CBDFM
El sistema ajusta la longitud del almacenamiento intermedio para devolver mensajes sin truncamiento.

Si no hay suficiente memoria disponible para asignar un almacenamiento intermedio para recibir el mensaje, el sistema llama a la función de devolución de llamada con un código de razón RC2071 .

Si, por ejemplo, solicita la conversión de datos y no hay suficiente memoria disponible para convertir los datos del mensaje, el mensaje no convertido se pasa a la función de devolución de llamada.

Este es un campo de entrada. El valor inicial del campo *MaxMsgLength* es CBDFM.

CBDOPT (entero con signo de 10 dígitos)

Estructura de descriptor de devolución de llamada-Campo Options.

Se puede especificar cualquiera de los siguientes, o todos ellos. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit). Se anotan las combinaciones que no son válidas; cualquier otra combinación es válida.

CBDFQ
La llamada MQCB falla si el gestor de colas está en estado de inmovilización.

En z/OS, esta opción también fuerza que la llamada MQCB falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

Especifique GMFIQ, en las opciones MQGMO que se pasan en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

Opciones de control: Las opciones siguientes controlan si se llama a la función de devolución de llamada, sin un mensaje, cuando cambia el estado del consumidor:

CDRC
La función de devolución de llamada se invoca con el tipo de llamada CBCTRC

.

CBDSC
La función de devolución de llamada se invoca con el tipo de llamada CBCTSC.

CBDTC
La función de devolución de llamada se invoca con el tipo de llamada CBCTTC.

CBDDC
La función de devolución de llamada se invoca con el tipo de llamada CBCTDC.

Consulte “CBCCALLT (entero con signo de 10 dígitos)” en la página 1041 para obtener más detalles sobre estos tipos de llamada.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

Nº CBD
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

CBDNO está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *Options* es CBDNO.

CBDSID (entero con signo de 10 dígitos)

Estructura del descriptor de devolución de llamada-Campo StrucId .

Este es el identificador de estructura; el valor debe ser:

CBDSI
Identificador de la estructura del descriptor de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es CBDSI.

CBDVER (entero con signo de 10 dígitos)

Estructura de descriptor de devolución de llamada-Campo de versión.

Este es el número de versión de la estructura; el valor debe ser:

CBDV1
Estructura del descriptor de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

CDBCV
Versión actual de la estructura del descriptor de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es CBDV1.

Valores iniciales

Tabla 688. Valores iniciales de los campos en MQCBD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	CBDSI	'CBD¬'
Version	CBDV1	1
CallBackType	CBTMC	1
Options	Nº CBD	0
CallbackArea	Ninguna	Bytes nulos
CallbackFunction	Ninguna	Bytes nulos
CallbackName	Ninguna	Espacios en blanco
MaxMsgLength	CBDFM	-1

Nota:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Serie de longitud variable) en IBM i

Utilice la estructura MQCHARV para describir una serie de longitud variable.

Visión general

Juego de caracteres y codificación: Los datos de MQCHARV deben estar en la codificación del gestor de colas local proporcionado por ENNAT y el juego de caracteres del campo VCHRC dentro de la estructura. Si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en la codificación del cliente. Algunos juegos de caracteres tienen una representación que depende de la codificación. Si VCHRC es uno de estos juegos de caracteres, la codificación utilizada es la misma que la de los otros campos de MQCHARV. El juego de caracteres identificado por VSCCSID puede ser un juego de caracteres de doble byte (DBCS).

Uso: la estructura MQCHARV direcciona los datos que pueden no estar contiguos con la estructura que los contiene. Para abordar estos datos, se pueden utilizar los campos declarados con el tipo de datos de puntero.

- "Campos" en la página 1053
- "Valores iniciales" en la página 1054
- "Declaración RPG" en la página 1054
- "Redefinición de CSAPL" en la página 1055

Campos

La estructura MQCHARV contiene los campos siguientes; los campos se describen en **orden alfabético**:

VCHRC (entero con signo de 10 dígitos)

Es el identificador de juego de caracteres de la serie de longitud variable dirigida por el campo VCHRP o VCHRO.

El valor inicial de este campo es CSAPL. Lo define IBM MQ para indicar que el gestor de colas debe cambiarlo por el identificador de juego de caracteres verdadero del gestor de colas. Esto es del mismo modo que se comporta CSQM. Como resultado, el valor CSAPL nunca se asocia con una serie de longitud variable. El valor inicial de este campo se puede cambiar definiendo un valor diferente para la constante CSAPL para la unidad de compilación mediante los medios adecuados para el lenguaje de programación de la aplicación.

VCHRL (entero con signo de 10 dígitos)

Longitud en bytes de la serie de longitud variable a la que se dirige el campo VCHRP o VCHRO.

El valor inicial de este campo es 0. El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

VSNLT

Si no se especifica VSNLT, los bytes VCHRL se incluyen como parte de la serie. Si hay caracteres nulos, no delimitan la serie.

Si se especifica VSNLT, la serie está delimitada por el primer nulo encontrado en la serie. El propio nulo no se incluye como parte de esa serie.

Nota: El carácter nulo utilizado para terminar una serie si se especifica VSNLT es un valor nulo del conjunto de códigos especificado por VCHRC.

Por ejemplo, en UTF-16 (CCSID 1200, 13488 y 17584), es la codificación Unicode de 2 bytes donde un valor nulo se representa mediante un número de 16 bits de todos los ceros. En UTF-16 es común encontrar bytes únicos establecidos en cero que forman parte de caracteres (por ejemplo, caracteres ASCII de 7 bits), pero las series sólo terminarán en nulo cuando se encuentren dos bytes 'cero' en un límite de bytes par. Es posible obtener dos 'cero' bytes en un límite impar cuando cada uno forma parte de caracteres válidos. Por ejemplo, x '01' x '00' x '00' x '30' representa dos caracteres Unicode válidos y no termina en nulo la serie.

VCHRO (entero con signo de 10 dígitos)

El desplazamiento en bytes de la serie de longitud variable desde el inicio de MQCHARV, o la estructura que la contiene.

Cuando la estructura MQCHARV está incorporada en otra estructura, este valor es el desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV. Cuando la estructura MQCHARV no está incorporada dentro de otra estructura, por ejemplo, si se especifica como parámetro en una llamada de función, el desplazamiento es relativo al inicio de la estructura MQCHARV.

El desplazamiento puede ser positivo o negativo. Puede utilizar el campo VCHRP o VCHRO para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es 0.

VCHRP (puntero)

Es un puntero a la serie de longitud variable.

Puede utilizar el campo VCHRP o VCHRO para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es un puntero nulo o bytes nulos.

VCHRS (entero con signo de 10 dígitos)

El tamaño en bytes del almacenamiento intermedio direccionado por el campo VCHRP o VCHRO.

Cuando se utiliza la estructura MQCHARV como campo de salida en una llamada a función, este campo debe inicializarse con la longitud del almacenamiento intermedio proporcionado. Si el valor de VCHRL es mayor que VCHRS, sólo se devolverán los bytes de datos VCHRS al interlocutor en el almacenamiento intermedio.

El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

VSUSL
Si se especifica VSUSL, la longitud del almacenamiento intermedio se obtiene del campo VCHRL de la estructura MQCHARV. Este valor especial no es adecuado cuando se utiliza la estructura como campo de salida y se proporciona un almacenamiento intermedio. Este es el valor inicial de este campo.

Valores iniciales

Tabla 689. Valores iniciales de MQCHARV para constantes

Nombre de campo	Nombre de constante	Valor de constante
VCHRP	Ninguna	Puntero nulo o bytes nulos.
VCHRO	Ninguna	0
VCHRS	VSUSL	-1
VCHRL	Ninguna	0
VCHRC	CSAPL	-3

Declaración 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
```

Redefinición de CSAPL

A diferencia de los lenguajes de programación soportados en otras plataformas, RPG no tiene una forma de redefinir una constante definida, por lo que debe establecer cada VCHRC específicamente si desea utilizar un valor que no sea CSAPL.

IBM i MQCIH (cabeceraCICS bridge) en IBM i

La estructura MQCIH describe la información que puede estar presente al principio de un mensaje enviado a CICS bridge a través de IBM MQ for z/OS.

Visión general

Nombre de formato: FMCICS.

Versión: La versión actual de MQCIH es CIVER2. Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQCIH, con el valor inicial del campo *CIVER* establecido en CIVER2.

Conjunto de caracteres y codificación: se aplican condiciones especiales al juego de caracteres y la codificación utilizados para la estructura MQCIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola CICS bridge deben proporcionar una estructura MQCIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQCIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQCIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; la conversión de MQCIH la realiza el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola CICS bridge .

  Nota: Hay una excepción a esto. Si el gestor de colas propietario de la cola CICS bridge utiliza CICS para la gestión de colas distribuidas, MQCIH debe estar en el juego de caracteres y la codificación del gestor de colas propietario de la cola CICS bridge .
- Los datos del mensaje de aplicación que siguen a la estructura MQCIH deben estar en el mismo juego de caracteres y codificación que la estructura MQCIH. Los campos *CICSI* y *CIENC* de la estructura MQCIH no se pueden utilizar para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

  El usuario debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

Uso: si los valores necesarios para la aplicación son los mismos que los valores iniciales que se muestran en la Tabla 691 en la página 1065, y el puente se ejecuta con AUTH=LOCAL o AUTH=IDENTIFY, la estructura MQCIH se puede omitir del mensaje. En todos los demás casos, la estructura debe estar presente.

El puente acepta una estructura MQCIH version-1 o version-2 , pero para las transacciones 3270 se debe utilizar una estructura version-2 .

La aplicación debe asegurarse de que los campos documentados como campos de "solicitud" tengan los valores adecuados en el mensaje enviado al puente; estos campos se introducen en el puente.

Los campos documentados como campos de "respuesta" los establece CICS bridge en el mensaje de respuesta que el puente envía a la aplicación. La información de error se devuelve en los campos *CIRET*,

CIFNC, *CICC*, *CIREA*y *CIAC* , pero no todos se establecen en todos los casos. La Tabla 690 en la página 1056 muestra qué campos se establecen para distintos valores de *CIRET*.

Tabla 690. Contenido de los campos de información de error en la estructura MQCIH

CIRET	***CIFNC***	***CICC***	***CIREA***	***CIAC***
CRC000	-	-	-	-
CRC003	-	-	FBC*	-
CRC002 CRC008	Nombre de llamada IBM MQ	IBM MQ *CMPCOD*	IBM MQ *REASON*	-
CRC001 CRC006 CRC007 CRC009	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	ABCODE de CICS

- "Campos" en la página 1056
- "Valores iniciales" en la página 1065
- "Declaración RPG" en la página 1066

Campos

La estructura MQCIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CIAC (serie de caracteres de 4 bytes)

Código de finalización anómala.

El valor devuelto en este campo sólo es significativo si el campo *CIRET* tiene el valor CRC005 o CRC004. Si lo hace, *CIAC* contiene el valor ABCODE de CICS .

Este es un campo de respuesta. La longitud de este campo la proporciona LNABNC. El valor inicial de este campo es de 4 caracteres en blanco.

Es un indicador que especifica si los descriptores ADS deben enviarse en las solicitudes SEND y RECEIVE BMS. Los valores siguientes están definidos:

ADNONE
No enviar ni recibir descriptor ADS.

ADSEND
Enviar descriptor ADS.

ADRECV
Recibir descriptor ADS.

ADMSGF
Utilice el formato de mensaje para el descriptor ADS.

Esto hace que el descriptor ADS se envíe o reciba utilizando el formato largo del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

El campo *CIADS* debe establecerse de la forma siguiente:

- Si no se están utilizando descriptores ADS, establezca el campo en ADNONE.
- Si se están utilizando los descriptores ADS y con el *mismo* CCSID en cada entorno, establezca el campo en la suma de ADSEND y ADRECV.
- Si se están utilizando los descriptores ADS , pero con CCSID *diferentes* en cada entorno, establezca el campo en la suma de ADSEND, ADRECV y ADMSGF.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es ADNONE.

CIADS (entero con signo de 10 dígitos)

Enviar/recibir descriptor ADS.

Es un indicador que especifica si los descriptores ADS deben enviarse en las solicitudes SEND y RECEIVE BMS. Los valores siguientes están definidos:

ADNONE
No enviar ni recibir descriptor ADS.

ADSEND
Enviar descriptor ADS.

ADRECV
Recibir descriptor ADS.

ADMSGF
Utilice el formato de mensaje para el descriptor ADS.

Esto hace que el descriptor ADS se envíe o reciba utilizando el formato largo del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

El campo *CIADS* debe establecerse de la forma siguiente:

- Si no se están utilizando descriptores ADS, establezca el campo en ADNONE.
- Si se están utilizando los descriptores ADS y con el *mismo* CCSID en cada entorno, establezca el campo en la suma de ADSEND y ADRECV.
- Si se están utilizando los descriptores ADS , pero con CCSID *diferentes* en cada entorno, establezca el campo en la suma de ADSEND, ADRECV y ADMSGF.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es ADNONE.

CIAI (serie de caracteres de 4 bytes)

Tecla AID.

Es el valor inicial de la tecla AID cuando se inicia la transacción. Es un valor de 1 byte, alineado a la izquierda.

Es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona LNATID. El valor inicial de este campo es de 4 blancos.

CIAUT (serie de caracteres de 8 bytes)

Contraseña o pase.

Se trata de una contraseña o passticket. Si la autenticación de identificador de usuario está activa para CICS bridge, se utiliza *CIAUT* con el identificador de usuario en el contexto de identidad MQMD para autenticar el remitente del mensaje.

Este es un campo de solicitud. La longitud de este campo la proporciona LNAUTH. El valor inicial de este campo es de 8 blancos.

CICC (entero con signo de 10 dígitos)

IBM MQ código de terminación o CICS EIBRESP.

El valor devuelto en este campo depende de *CIRET* ; consulte Tabla 690 en la página 1056.

Este es un campo de respuesta. El valor inicial de este campo es CCOK.

CICNC (serie de caracteres de 4 bytes)

Código de transacción de terminación anómala.

Este es el código de terminación anómala que se debe utilizar para terminar la transacción (normalmente una transacción conversacional que solicita más datos). De lo contrario, este campo se establece en blancos.

Es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona LNCNCL. El valor inicial de este campo es de 4 blancos.

CICP (entero con signo de 10 dígitos)

Posición del cursor.

Es la posición inicial del cursor cuando se inicia la transacción. Posteriormente, para las transacciones conversacionales, la posición del cursor está en el vector RECEIVE.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *CIVER* es menor que CIVER2.

CICSI (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

CICT (entero con signo de 10 dígitos)

Si la tarea puede ser conversacional.

Se trata de un indicador que especifica si se debe permitir que la tarea emita solicitudes para obtener más información o si se debe terminar de forma anómala. El valor debe ser uno de los siguientes:

SÍ
La tarea es conversacional.

CNO
La tarea no es conversacional.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es CTNO.

CIENC (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

CIEO (entero con signo de 10 dígitos)

Desplazamiento del error en el mensaje.

Esta es la posición de los datos no válidos detectados por la salida del puente. Este campo proporciona el desplazamiento desde el inicio del mensaje hasta la ubicación de los datos no válidos.

Se trata de un campo de respuesta utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *CIVER* es menor que CIVER2.

CIFAC (serie de bits de 8 bytes)

Señal de recurso de puente.

Se trata de una señal de recurso de puente de 8 bytes. La finalidad de una señal de recurso de puente es permitir que varias transacciones de una pseudoconversación utilicen el mismo recurso de puente (terminal 3270 virtual). En el primer, o sólo, mensaje de una pseudoconversación, se debe establecer un valor de FCNONE; esto indica a CICS que asigne un nuevo recurso de puente para este mensaje. Se devuelve una señal de recurso de puente en los mensajes de respuesta cuando se especifica un *CIFKT* distinto de cero en el mensaje de entrada. A continuación, los mensajes de entrada posteriores pueden utilizar la misma señal de recurso de puente.

Se define el siguiente valor especial:

FCNONE
No se ha especificado ninguna señal BVT.

Se trata tanto de un campo de solicitud como de un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona LNFAC. El valor inicial de este campo es FCNONE.

CIFKT (entero con signo de 10 dígitos)

Tiempo de liberación de recurso de puente.

Es la cantidad de tiempo en segundos que se conservará el recurso de puente después de que haya finalizado la transacción de usuario. Para transacciones no conversacionales, el valor debe ser cero.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0.

CIFL (serie de caracteres de 4 bytes)

Atributos emulados de terminal.

Es el nombre de un terminal instalado que se va a utilizar como modelo para el recurso de puente. Un valor de blancos significa que *CIFL* se toma de la definición de perfil de transacción de puente o que se utiliza un valor por omisión.

Es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona LNFACL. El valor inicial de este campo es de 4 blancos.

CIFLG (entero con signo de 10 dígitos)

Distintivos.

El valor debe ser:

CIFNON
Sin distintivos.

Este es un campo de solicitud. El valor inicial de este campo es CIFNON.

CIFMT (serie de caracteres de 8 bytes)

Nombre de formato IBM MQ de los datos que siguen a MQCIH.

Especifica el nombre de formato IBM MQ de los datos que siguen a la estructura MQCIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

Este nombre de formato también se utiliza para el mensaje de respuesta, si el campo *CIRFM* tiene el valor FMNONE.

- Para las solicitudes DPL, *CIFMT* debe ser el nombre de formato de COMMAREA.
- Para las solicitudes 3270, *CIFMT* debe ser CSQCBDCIy *CIRFM* debe ser CSQCBDCO.

Las salidas de conversión de datos para estos formatos deben estar instaladas en el gestor de colas donde se van a ejecutar.

Si el mensaje de solicitud da como resultado la generación de un mensaje de respuesta de error, el mensaje de respuesta de error tiene un nombre de formato de FMSTR.

Este es un campo de solicitud. La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

CIFNC (serie de caracteres de 4 bytes)

Nombre de llamada IBM MQ o función EIBFN de CICS .

El valor devuelto en este campo depende de *CIRET* ; consulte Tabla 690 en la página 1056. Los valores siguientes son posibles cuando *CIFNC* contiene un nombre de llamada IBM MQ :

CFCONN
Llamada MQCONN.

CGET
Llamada MQGET.

CFINQ
Llamada MQINQ.

FOPEN
Llamada MQOPEN.

CPUT
Llamada MQPUT.

CFPUT1
Llamada MQPUT1 .

CFNONE
Sin llamada.

Este es un campo de respuesta. La longitud de este campo la da LNFUNC. El valor inicial de este campo es CFNONE.

CIGWI (entero con signo de 10 dígitos)

Intervalo de espera para la llamada MQGET emitida por la tarea de puente.

Este campo sólo es aplicable cuando *CIUOW* tiene el valor CUFRST. Permite a la aplicación emisora especificar el tiempo aproximado en milisegundos que las llamadas MQGET emitidas por el puente deben esperar a los segundos y posteriores mensajes de solicitud para la unidad de trabajo iniciada por este mensaje. Esto altera temporalmente el intervalo de espera predeterminado utilizado por el puente. Se pueden utilizar los siguientes valores especiales:

WIDFLT
Intervalo de espera predeterminado.

Esto hace que el CICS bridge espere el periodo especificado cuando se inició el puente.

WIULIM
Intervalo de espera ilimitado.

Este es un campo de solicitud. El valor inicial de este campo es WIDFLT.

CIII (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado. El valor debe ser 0. Este campo no está presente si *CIVER* es menor que CIVER2.

CILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQCIH.

El valor debe ser uno de los siguientes:

CILEN1
Longitud de la estructura de cabecera de información version-1 CICS .

CILEN2
Longitud de la estructura de cabecera de información de version-2 CICS .

La constante siguiente especifica la longitud de la versión actual:

CILENC
Longitud de la versión actual de la estructura de cabecera de información de CICS .

Este es un campo de solicitud. El valor inicial de este campo es CILEN2.

CILT (entero con signo de 10 dígitos)

Tipo de enlace.

Indica el tipo de objeto que el puente debe intentar enlazar. El valor debe ser uno de los siguientes:

LTPROG
Programa DPL.

LTTRAN
Transacción 3270.

Este es un campo de solicitud. El valor inicial de este campo es LTPROG.

CINTI (serie de caracteres de 4 bytes)

Siguiente transacción a adjuntar.

Es el nombre de la siguiente transacción devuelta por la transacción de usuario (normalmente por EXEC CICS RETURN TRANSID). Si no hay ninguna transacción siguiente, este campo se establece en blancos.

Se trata de un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona LNTRID. El valor inicial de este campo es de 4 blancos.

CIODL (entero con signo de 10 dígitos)

Longitud de datos COMMAREA de salida.

Es la longitud de los datos de usuario que se van a devolver al cliente en un mensaje de respuesta. Esta longitud incluye el nombre de programa de 8 bytes. La longitud de COMMAREA pasada al programa enlazado es el máximo de este campo y la longitud de los datos de usuario en el mensaje de solicitud, menos 8.

Nota: La longitud de los datos de usuario en un mensaje es la longitud del mensaje *excluyendo* la estructura MQCIH.

Si la longitud de los datos de usuario en el mensaje de solicitud es menor que *CIODL*, se utiliza la opción DATALENGTH del mandato LINK ; esto permite que LINK se envíe de forma eficaz a otra región CICS .

Se puede utilizar el siguiente valor especial:

OLINPT
La longitud de salida es la misma que la longitud de entrada.

Este valor puede ser necesario incluso si no se solicita ninguna respuesta, para asegurarse de que el COMMAREA pasado al programa enlazado tiene un tamaño suficiente.

Este es un campo de petición utilizado sólo para programas DPL. El valor inicial de este campo OLINPT.

CIREA (entero con signo de 10 dígitos)

IBM MQ código de razón o comentarios, o CICS EIBRESP2.

El valor devuelto en este campo depende de *CIRET* ; consulte Tabla 690 en la página 1056.

Este es un campo de respuesta. El valor inicial de este campo es RCNONE.

CIRET (entero con signo de 10 dígitos)

Código de retorno del puente.

Este es el código de retorno del CICS bridge que describe el resultado del proceso realizado por el puente. Los campos *CIFNC*, *CICC*, *CIREA*y *CIAC* pueden contener información adicional (consulte Tabla 690 en la página 1056). El valor puede ser uno de los siguientes:

CRC000
(0, X'000 ') Sin error.

CRC001
(1, X'001 ') La sentencia EXEC CICS ha detectado un error.

CRC002
(2, X'002 ') La llamada IBM MQ ha detectado un error.

CRC003
(3, X'003 ') CICS bridge ha detectado un error.

CRC004
(4, X'004 ') CICS bridge ha finalizado de forma anómala.

CRC005
(5, X'005 ') La aplicación ha finalizado de forma anómala.

CRC006
(6, X'006 ') Se ha producido un error de seguridad.

CRC007
(7, X'007 ') Programa no disponible.

CRC008
(8, X'008 ') Segundo o posterior mensaje dentro de la unidad de trabajo actual no recibido dentro del tiempo especificado.

CRC009
(9, X'009 ') Transacción no disponible.

Este es un campo de respuesta. El valor inicial de este campo es CRC000.

CIRFM (serie de caracteres de 8 bytes)

Nombre de formato IBM MQ del mensaje de respuesta.

Es el nombre de formato IBM MQ del mensaje de respuesta que se enviará en respuesta al mensaje actual. Las reglas para codificarlo son las mismas que para el campo *MDFMT* en MQMD.

Este es un campo de petición utilizado sólo para programas DPL. La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

CIRSI (serie de caracteres de 4 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 4 blancos. La longitud de este campo la proporciona LNRSID.

CIRS1 (serie de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS2 (serie de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS3 (serie de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS4 (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado. El valor debe ser 0. Este campo no está presente si *CIVER* es menor que CIVER2.

CIRTI (serie de caracteres de 4 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 4 blancos. La longitud de este campo la proporciona LNTRID.

CISC (serie de caracteres de 4 bytes)

Código de inicio de transacción.

Es un indicador que especifica si el puente emula una transacción de terminal o una transacción START. El valor debe ser uno de los siguientes:

SCSTRT
Inicio.

SCDATA
Datos de inicio.

SCTERM
Terminar entrada.

SCNONA
Ninguno.

En la respuesta del puente, este campo se establece en el código de inicio adecuado para el siguiente ID de transacción contenido en el campo *CINTI* . Los siguientes códigos de inicio son posibles en la respuesta:

- SCSTRT
- SCDATA
- SCTERM

Para CICS Transaction Server 1.2, este campo es sólo un campo de solicitud; su valor en la respuesta no está definido.

Para CICS Transaction Server 1.3 y releases posteriores, se trata de un campo de solicitud y de respuesta.

Este campo sólo se utiliza para transacciones 3270. La longitud de este campo la proporciona LNSTCO. El valor inicial de este campo es SCNONE.

CISID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

CISIDV
Identificador de la estructura de cabecera de información de CICS .

Este es un campo de solicitud. El valor inicial de este campo es CISIDV.

CITES (entero con signo de 10 dígitos)

Estado al final de la tarea.

Este campo muestra el estado de la transacción de usuario al final de la tarea. Se devuelve uno de los siguientes valores:

TENOSY
No sincronizado.

La transacción de usuario todavía no se ha completado y no se ha sincronizado. El campo *MDMT* en MQMD es MTRQST en este caso.

TECMIT
Confirmar unidad de trabajo.

La transacción de usuario todavía no se ha completado, pero ha sincronizado la primera unidad de trabajo. El campo *MDMT* en MQMD es MTDGRM en este caso.

TEBACK
Restituir unidad de trabajo.

La transacción de usuario todavía no se ha completado. La unidad de trabajo actual se restituirá. El campo *MDMT* en MQMD es MTDGRM en este caso.

TEENDT
Finalizar tarea.

La transacción de usuario ha finalizado (o ha terminado de forma anómala). El campo *MDMT* en MQMD es MTRPLY en este caso.

Se trata de un campo de respuesta utilizado sólo para transacciones 3270. El valor inicial de este campo es TENOSY.

CITI (serie de caracteres de 4 bytes)

Transacción a adjuntar.

Si *CILT* tiene el valor LTTRAN, *CITI* es el identificador de transacción de la transacción de usuario que se va a ejecutar; en este caso debe especificarse un valor no en blanco.

Si *CILT* tiene el valor LTPROG, *CITI* es el código de transacción bajo el que se van a ejecutar todos los programas de la unidad de trabajo. Si el valor especificado está en blanco, se utiliza el código de transacción predeterminado del puente CICS DPL (CKBP). Si el valor no está en blanco, debe haberse definido en CICS como una TRANSACTION local con un programa inicial de CSQCBP00. Este campo sólo es aplicable cuando *CIUOW* tiene el valor CUFRST o CUONLY.

Este es un campo de solicitud. La longitud de este campo la proporciona LNTRID. El valor inicial de este campo es de 4 blancos.

CIUOW (entero con signo de 10 dígitos)

Control de unidad de trabajo.

Esto controla el proceso de unidad de trabajo realizado por el CICS bridge. Puede solicitar al puente que ejecute una sola transacción o uno o varios programas dentro de una unidad de trabajo. El campo indica si el CICS bridge debe iniciar una unidad de trabajo, realizar la función solicitada dentro de la unidad de trabajo actual o finalizar la unidad de trabajo comprometiéndola o retrotrayéndola. Se soportan varias combinaciones, para optimizar los flujos de transmisión de datos.

El valor debe ser uno de los siguientes:

CUONLY
Inicie la unidad de trabajo, realice la función y, a continuación, confirme la unidad de trabajo (DPL y 3270).

CUCONT
Datos adicionales para la unidad de trabajo actual (sólo 3270).

CUFRST
Inicie la unidad de trabajo y realice la función (sólo DPL).

CUMIDL
Realizar función dentro de la unidad de trabajo actual (sólo DPL).

CULOS
Realice la función y, a continuación, confirme la unidad de trabajo (sólo DPL).

CUCMIT
Confirme la unidad de trabajo (sólo DPL).

CUBACK
Restituir la unidad de trabajo (sólo DPL).

Este es un campo de solicitud. El valor inicial de este campo es CUONLY.

CIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

CIVER1
Version-1 CICS estructura de cabecera de información.

CIVER2
Version-2 CICS estructura de cabecera de información.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

CIVERC
Versión actual de la estructura de cabecera de información de CICS .

Este es un campo de solicitud. El valor inicial de este campo es CIVER2.

Valores iniciales

Tabla 691. Valores iniciales de los campos en MQCIH

Nombre de campo	Nombre de constante	Valor de constante
CISID	CISIDV	'CIH¬'
CIVER	CIVER2	2
CILEN	CILEN2	180
CIENC	Ninguna	0
CICSI	Ninguna	0
CIFMT	FMNONE	Espacios en blanco
CIFLG	CIFNON	0
CIRET	CRC000	0
CICC	CCOK	0
CIREA	RCNONE	0
CIUOW	CUONLY	273
CIGWI	WIDFLT	-2
CILT	LTPROG	1
CIODL	OLINPT	-1
CIFKT	Ninguna	0
CIADS	ADNONE	0
CICT	CNO	0
CITES	TENOSY	0
CIFAC	FCNONE	Nulos
CIFNC	CFNONE	Espacios en blanco
CIAC	Ninguna	Espacios en blanco
CIAUT	Ninguna	Espacios en blanco
CIRS1	Ninguna	Espacios en blanco
CIRFM	FMNONE	Espacios en blanco
CIRSI	Ninguna	Espacios en blanco
CIRTI	Ninguna	Espacios en blanco
CITI	Ninguna	Espacios en blanco
CIFL	Ninguna	Espacios en blanco
CIAI	Ninguna	Espacios en blanco
CISC	SCNONA	Espacios en blanco

Tabla 691. Valores iniciales de los campos en MQCIH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
CICNC	Ninguna	Espacios en blanco
CINTI	Ninguna	Espacios en blanco
CIRS2	Ninguna	Espacios en blanco
CIRS3	Ninguna	Espacios en blanco
CICP	Ninguna	0
CIEO	Ninguna	0
CIII	Ninguna	0
CIRS4	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Crear opciones de manejador de mensajes) en IBM i

La estructura **MQCMHO** permite a las aplicaciones especificar opciones que controlan cómo se crean los manejadores de mensajes.

Visión general

Finalidad

La estructura es un parámetro de entrada en la llamada **MQCRTMH** .

Juego de caracteres y codificación

Los datos de **MQCMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1067
- “Valores iniciales” en la página 1069
- “Declaración RPG” en la página 1069

Campos

La estructura MQCMHO contiene los campos siguientes; los campos se describen en orden alfabético:

CMOPT (entero con signo de 10 dígitos)

Se puede especificar una de las opciones siguientes:

CMVAL

Cuando se llama a **MQSETMP** para establecer una propiedad en este descriptor de mensaje, el nombre de propiedad se valida para asegurarse de que:

- no contiene caracteres no válidos.
- no empieza por "JMS" o "usr.JMS" excepto para lo siguiente:
  - JMSCorrelationID
  - JMSReplyTo

- JMSType
- JMSXGroupID
- JMSXGroupSeq

Estos nombres están reservados para las propiedades JMS .

- no es una de las palabras clave siguientes, en cualquier mezcla de mayúsculas o minúsculas:
  - "Y"
  - "ENTRE"
  - "ESCAPE"
  - "FALSO"
  - "IN"
  - "IS"
  - "like"
  - "not"
  - "NULO"
  - "O"
  - "VERDADERO"
- no empieza "Cuerpo." o "Root." (excepto para "Root.MQMD.").

Si la propiedad es MQ-defined ("mq. *") y el nombre se reconoce, los campos de descriptor de propiedad se establecen en los valores correctos para la propiedad. Si la propiedad no se reconoce, el campo *Support* del descriptor de propiedad se establece en **PDSUPO** (para obtener más información, consulte PDSUP).

CMDEFV

Especifica que se produce el nivel predeterminado de validación de los nombres de propiedad.

El nivel predeterminado de validación es equivalente al especificado por **CMVAL**.

En un futuro release, se puede definir una opción administrativa que cambiará el nivel de validación que se producirá cuando se defina **CMDEFV** .

Éste es el valor predeterminado.

CMNOVA

No se produce ninguna validación en el nombre de propiedad. Consulte la descripción de **CMVAL**.

Opción predeterminada: Si no se requiere ninguna de las opciones descritas anteriormente en esta sección, se puede utilizar la opción siguiente:

CMNONE

Todas las opciones asumen sus valores predeterminados. Utilice este valor para indicar que no se ha especificado ninguna otra opción. **CMNONE** ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Siempre es un campo de entrada. El valor inicial de este campo es **CMDEFV**.

CMSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

CMSIDV
Identificador para la estructura de opciones de creación de manejadores de mensajes.

Siempre es un campo de entrada. El valor inicial de este campo es **CMSIDV**.

CMVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

CMVER1
Version-1 crea la estructura de opciones de manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

CMVERC
Versión actual de la estructura de opciones de creación de manejadores de mensajes.

Siempre es un campo de entrada. El valor inicial de este campo es **CMVER1**.

Valores iniciales

Tabla 692. Valores iniciales de los campos en MQCMHO

Nombre de campo	Nombre de constante	Valor de constante
CMSID	CMSIDV	'CMHO'
CMVER	CMVER1	1
CMOPT	CMDEFV	0

Declaración 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 (opciones de conexión) en IBM i

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión con el gestor de colas local.

Visión general

Finalidad: la estructura es un parámetro de entrada/salida en la llamada MQCONNX.

Versión: La versión actual de MQCNO es CNVER6. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQCNO soportada por el entorno, pero con el valor inicial del campo CNVER establecido en CNVER1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo CNVER en el número de versión de la versión necesaria.

Conjunto de caracteres y codificación: los datos de MQCNO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

- “Campos” en la página 1070

- “Valores iniciales” en la página 1075
- “Declaración RPG” en la página 1075

Campos

La estructura MQCNO contiene los campos siguientes; los campos se describen en **orden alfabético**:

CCDTUL (entero con signo de 10 dígitos)

CCDTUL es la longitud de la serie identificada por CCDTUP o CCDTUO que contiene un URL que identifica la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión.

Utilice CCDTUL sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como IBM MQ MQI client.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora CCDTUL.

Este campo se ignora si CNVER es menor que CNVER6.

CCDTUO (entero con signo de 10 dígitos)

CCDTUO es el desplazamiento en bytes, desde el inicio de la estructura MQCNO, a una serie que contiene un URL que identifica la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión. El desplazamiento puede ser positivo o negativo.

Utilice CCDTUL sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como IBM MQ MQI client.

Importante: Solo puede utilizar uno de CCDTUP y CCDTUO. La llamada falla con el código de razón RC2600 si ambos campos son distintos de cero.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora CCDTUO.

Este campo se ignora si CNVER es menor que CNVER6.

CCDTUP (puntero)

CCDTUP es un puntero opcional a una serie que contiene un URL, para identificar la ubicación de la tabla de canales de conexión de cliente que se debe utilizar para la conexión.

Utilice CCDTUP sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como IBM MQ MQI client.

Importante: Solo puede utilizar uno de CCDTUP y CCDTUO. La llamada falla con el código de razón RC2600 si ambos campos son distintos de cero.

Esta es una alternativa programática a establecer las variables de entorno MQCHLLIB y MQCHLTAB .

Si la aplicación no se ejecuta como un cliente, se ignora CCDTUP.

Este campo se ignora si CNVER es menor que CNVER6.

CNCCO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes de una estructura de definición de canal MQCD desde el inicio de la estructura MQCNO.

CNCCP (puntero)

Es un puntero a una estructura de definición de canal MQCD.

CNCONID (serie de caracteres de 24 bytes)

Identificador de conexión exclusivo. Este campo permite al gestor de colas identificar de forma fiable un proceso de aplicaciones asignándole un identificador exclusivo cuando se conecta por primera vez al gestor de colas.

Las aplicaciones utilizan el identificador de conexión para fines de correlación al realizar llamadas PUT y GET. El gestor de colas asigna un identificador a todas las conexiones, independientemente de cómo se haya establecido la conexión.

Es posible utilizar el identificador de conexión para forzar el final de una unidad de trabajo de larga ejecución. Para ello, especifique el identificador de conexión utilizando el mandato PCF 'Detener conexión' o el mandato MQSC STOP CONN. Para obtener más información sobre cómo utilizar estos mandatos, consulte los enlaces relacionados.

El valor inicial del campo es de 24 bytes nulos.

CNCT (serie de 128 bytes)

Esta es una etiqueta que el gestor de colas asocia con los recursos afectados por la aplicación durante esta conexión.

Etiqueta de conexión del gestor de colas.

Cada aplicación o instancia de aplicación debe utilizar un valor diferente para la etiqueta, para que el gestor de colas pueda serializar correctamente el acceso a los recursos afectados. Consulte las descripciones de las opciones CN* CT* para obtener más detalles. La etiqueta deja de ser válida cuando la aplicación termina o emite la llamada MQDISC.

Utilice el siguiente valor especial si no se necesita ninguna etiqueta:

CTNONE
No se ha especificado ninguna etiqueta de conexión.

El valor es cero binario para la longitud del campo.

Este es un campo de entrada. La longitud de este campo la proporciona LNCTAG. El valor inicial de este campo es CTNONE. Este campo se ignora si CNVER es menor que CNVER3.

Utilice el campo ConnTag al conectarse a un gestor de colas de z/OS .

CNOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQCONNX.

Opciones de enlace

Las opciones de enlace controlan el tipo de enlace IBM MQ que se utiliza; especifique sólo una de estas opciones:

CNSBND
Enlace estándar.

La opción de enlace estándar hace que la aplicación y el agente del gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas errantes.

Utilice CNSBND en situaciones en las que es posible que la aplicación no se haya probado completamente, o que no sea fiable o no sea fiable. CNSBND es el valor predeterminado.

CNSBND se define para ayudar a la documentación del programa. No utilice esta opción con ninguna otra opción que controle el tipo de enlace utilizado; pero debido a que su valor es cero, no se puede detectar dicho uso.

Esta opción está soportada en todos los entornos.

CNFBND
Enlace de vía de acceso rápida.

La opción de enlace de vía de acceso rápida hace que la aplicación y el agente del gestor de colas local formen parte de la misma unidad de ejecución. La vía de acceso rápida contrasta con el enlace estándar, donde la aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes.

CNFBND se ignora si el gestor de colas no da soporte a este tipo de enlace; el proceso continúa como si no se hubiera especificado la opción.

CNFBND puede ser ventajoso en situaciones en las que varios procesos consumen más recursos que el recurso global utilizado por la aplicación.Una aplicación que utiliza el enlace de vía de acceso rápida se conoce como *aplicación de confianza*.

Tenga en cuenta los siguientes puntos importantes a la hora de decidir si se debe utilizar el enlace de vía de acceso rápida:

- **La utilización de la opción CNFBND no impide que una aplicación altere o dañe mensajes y otras áreas de datos pertenecientes al gestor de colas. Utilice esta opción sólo en situaciones en las que haya evaluado completamente estos problemas.**
- La aplicación no debe utilizar señales asíncronas o interrupciones de temporizador (como sigkill) con CNFBND. También hay restricciones sobre el uso de segmentos de memoria compartida.
- La aplicación no debe tener más de una hebra conectada al gestor de colas a la vez.
- La aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
- La aplicación debe finalizar antes de finalizar el gestor de colas con el mandato endmqm .

Los puntos siguientes se aplican al uso de CNFBND en los entornos indicados:

- En IBM i, el trabajo debe ejecutarse bajo el perfil de usuario QMQM que pertenece al grupo QMQMADM . Además, el programa no debe terminar de forma anómala, de lo contrario podrían producirse resultados imprevisibles.

Para obtener más información sobre las implicaciones de utilizar aplicaciones de confianza, consulte Conexión a un gestor de colas utilizando la llamada MQCONNX y Restricciones para aplicaciones de confianza.

CNSHBD

Enlaces compartidos.

La opción de enlaces compartidos hace que la aplicación y el agente del gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas errantes. Sin embargo, algunos recursos se comparten entre la aplicación y el agente del gestor de colas local. CNSHBD se ignora si el gestor de colas no da soporte a este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

CNIBND

Enlaces aislados.

La opción de enlaces aislados hace que la aplicación y el agente del gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas errantes. El proceso de aplicaciones y el agente del gestor de colas local se aíslan entre sí en el sentido de que no comparten recursos. CNIBND se ignora si el gestor de colas no da soporte a este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

Opciones de manejo compartido

Las opciones siguientes controlan la compartición de descriptores de contexto entre distintas hebras (unidades de proceso paralelo) dentro del mismo proceso. Sólo se puede especificar una de estas opciones.

CNHSN
No hay compartimiento de manejadores entre hebras.

La opción de no compartición de descriptores de contexto entre hebras indica que la conexión y los descriptores de contexto de objeto sólo pueden ser utilizados por la hebra que ha hecho que se asigne el descriptor de contexto; es decir, la hebra que ha emitido la llamada `MQCONN`, `MQCONNX`o `MQOPEN` . Otras hebras que pertenecen al mismo proceso no pueden utilizar los descriptores de contexto.

CNHSB
Compartición de descriptores de contexto serie entre hebras, con bloqueo de llamadas.

La opción de compartición de descriptores de contexto serie entre hebras, con bloqueo de llamadas, indica que los descriptores de contexto de conexión y objeto asignados por una hebra de un proceso pueden ser utilizados por otras hebras pertenecientes al mismo proceso. Sin embargo, sólo una hebra a la vez puede utilizar un descriptor de contexto determinado, es decir, sólo se permite el uso en serie de un descriptor de contexto. Si una hebra intenta utilizar un descriptor de contexto que ya está en uso por otra hebra, la llamada se bloquea (espera) hasta que el descriptor de contexto esté disponible.

CNHSNB
Compartición de descriptores de contexto serie entre hebras, sin bloqueo de llamadas.

La opción de compartición de descriptores de contexto serie entre hebras, sin bloqueo de llamadas, es la misma que la opción " *con la opción* blocando ", excepto que, si el descriptor de contexto está siendo utilizado por otra hebra, la llamada se completa inmediatamente con `CCFAIL` y `RC2219` en lugar de bloquearse hasta que el descriptor de contexto pase a estar disponible.

Una hebra puede tener cero o un descriptor de contexto no compartido, más cero o más descriptores de contexto compartidos:

- Cada llamada `MQCONN` o `MQCONNX` que especifica `CNHSN` devuelve un nuevo descriptor de contexto no compartido en la primera llamada, y el mismo descriptor de contexto no compartido en las llamadas posteriores (suponiendo que no haya ninguna llamada `MQDISC` interviniente). El código de razón es `RC2002` para la segunda llamada y posteriores.
- Cada llamada `MQCONNX` que especifica `CNHSB` o `CNHSNB` devuelve un nuevo descriptor de contexto compartido en cada llamada.

Los descriptores de contexto de objeto heredan las mismas propiedades de compartición que el descriptor de contexto de conexión especificado en la llamada `MQOPEN` que ha creado el descriptor de contexto de objeto. Además, las unidades de trabajo heredan las mismas propiedades de compartición que el descriptor de conexión utilizado para iniciar la unidad de trabajo; si la unidad de trabajo se inicia en una hebra utilizando un descriptor de contexto compartido, la unidad de trabajo se puede actualizar en otra hebra utilizando el mismo descriptor de contexto.

Si no especifica una opción de uso compartido de descriptor de contexto, el valor predeterminado lo determina el entorno:

- En el entorno de Microsoft Transaction Server (MTS), el valor predeterminado es el mismo que `CNHSB`.
- En otros entornos, el valor predeterminado es el mismo que `CNHSN`.

Opciones de reconexión

Las opciones de reconexión determinan si una conexión es reconectable. Sólo se pueden volver a conectar las conexiones de cliente.

CNRCDF
La opción de reconexión se resuelve en su valor predeterminado. Si no se establece ningún valor predeterminado, el valor de esta opción se resuelve en `DISABLED`. El valor de la opción se pasa al servidor y **PCF** y **MQSC**pueden consultarlo.

CNRC
La aplicación se puede reconectar a cualquier gestor de colas coherente con el valor del parámetro MQCONNX **QMNAME** . Utilice la opción CNRC sólo si no hay afinidad entre la aplicación cliente y el gestor de colas con el que estableció inicialmente una conexión. El valor de la opción se pasa al servidor y **PCF** y **MQSC**pueden consultarlo.

CNRCD
La aplicación no se puede volver a conectar. El valor de la opción no se pasa al servidor.

CNRCQM
La aplicación sólo se puede reconectar al gestor de colas con el que se conectó originalmente. Utilice este valor si un cliente se puede reconectar, pero existe una afinidad entre la aplicación cliente y el gestor de colas con el que estableció originalmente una conexión. Elija este valor si desea que el cliente se reconecte automáticamente a la instancia en espera de un gestor de colas con elevada disponibilidad. El valor de la opción se pasa al servidor y **PCF** y **MQSC**pueden consultarlo.

Utilice las opciones CNRC, CNRCDy CNRCQM sólo para conexiones de cliente. Si las opciones se utilizan para una conexión de enlace, MQCONNX falla con el código de terminación, MQCC_FAILED y el código de razón, MQRC_OPTIONS_ERROR.

Opción predeterminada: Si ninguna de las opciones descritas es necesaria, se puede utilizar la opción siguiente:

NINGUNO
No se especifica ninguna opción.

CNNONE se define para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción CN* , pero debido a que su valor es cero, este uso no se puede detectar.

CNSCO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes de una estructura MQSCO desde el inicio de la estructura MQCNO.

Este campo se ignora si CNVER es menor que CNVER4.

CNSCP (puntero)

Es la dirección de una estructura MQSCO.

Este campo se ignora si CNVER es menor que CNVER4.

CNSECPO (entero con signo de 10 dígitos)

Desplazamiento de parámetros de seguridad. El desplazamiento de la estructura MQCSP utilizada para especificar un ID de usuario y una contraseña.

El valor puede ser positivo o negativo. El valor inicial de este campo es 0.

Este campo se ignora si CNVER es menor que CNVER5.

CNSECPP (puntero)

Puntero de parámetros de seguridad. Dirección de la estructura MQCSP utilizada para especificar un ID de usuario y una contraseña.

El valor inicial de este campo es un puntero nulo o bytes nulos.

Este campo se ignora si CNVER es menor que CNVER5.

CNSID (serie de caracteres de 4 bytes)

Identificador de estructura para la estructura MQCNO.

El valor debe ser:

CNSIDV
Identificador de la estructura de opciones de conexión.

Siempre es un campo de entrada. El valor inicial de este campo es CNSIDV.

CNVER (entero con signo de 10 dígitos)

El número de versión de estructura para la estructura MQCNO.

El valor debe ser:

CNVER6
Estructura de opciones de conexión de Version-6 .

Esta versión está soportada en todos los entornos.

V 9.1.2 **CNVER7**
Estructura de opciones de conexión de Version-7 .

Esta versión está soportada en todos los entornos.

La constante siguiente especifica el número de versión de la versión actual:

CNVERC
Versión actual de la estructura de opciones de conexión.

V 9.1.2 Siempre es un campo de entrada. El valor inicial de este campo es CNVER7.

Valores iniciales

Tabla 693. Valores iniciales de los campos en MQCNO

Nombre de campo	Nombre de constante	Valor de constante
CNSID	CNSIDV	'CNO¬
CNVER	CNVER5	1
CNOPT	CNNONE	0
CNCCO	Ninguna	0
CNCCP	Ninguna	Puntero nulo o bytes nulos
CNCT	CTNONE	Nulos
CNSCP	Ninguna	Puntero nulo o bytes nulos
CNSCO	Ninguna	0
CNCONID	Ninguna	Nulos
CNSECPO	Ninguna	0
CNSECPP	Ninguna	Puntero nulo o bytes nulos
CCDTUL	Ninguna	0
CCDTUO	Ninguna	0
CCDTUP	Ninguna	Puntero nulo o bytes nulos

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración RPG

```
D*****************************************************************
    D**                                                           **
    D**                    IBM MQ for IBM i                       **
    D**                                                           **
    D**  FILE NAME:      CMQCNOG                                  **
    D**                                                           **
```

```
D**  DESCRIPTION:    MQCNO Structure -- Connect Options          **
D**                                                              **
D****************************************************************
D**  <N_OCO_COPYRIGHT>                                           **
D**  Licensed Materials - Property of IBM                        **
D**                                                              **
D**  5724-H72                                                    **
D**  (c) Copyright IBM Corp. 1993, 2024. All Rights Reserved.    **
D**                                                              **
D**  US Government Users Restricted Rights - Use, duplication or **
D**  disclosure restricted by GSA ADP Schedule Contract with     **
D**  IBM Corp.                                                   **
D**  <NOC_COPYRIGHT>                                             **
D****************************************************************
D**                                                              **
D**  FUNCTION:       This file declares the structure MQCNO,     **
D**                  which is used by the main MQI.              **
D**                                                              **
D**  PROCESSOR:      RPG (ILE)                                   **
D**                                                              **
D****************************************************************
D*
D*
D****************************************************************
D** <BEGIN_BUILDINFO>                                            **
D** Generated on:  08/02/16 13:50                                **
D** Build Level:   L000000                                       **
D** Build Type:    Production                                    **
D** Pointer Size:  128 Bit                                       **
D** Source File:                                                 **
D** CMQCNOG                                                      **
D** <END_BUILDINFO>                                              **
D****************************************************************
D*
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D*
D* MQCNO Structure
D*
D* Structure identifier
D  CNSID                  1      4    INZ('CNO ')
D* Structure version number
D  CNVER                  5      8I 0 INZ(1)
D* Options that control the action of MQCONNX
D  CNOPT                  9     12I 0 INZ(0)
D* Ver:1 **
D* Offset of MQCD structure for client connection
D  CNCCO                 13     16I 0 INZ(0)
D* Address of MQCD structure for client connection
D  CNCCP                 17     32*   INZ(*NULL)
D* Ver:2 **
D* Queue managerconnection tag
D  CNCT                  33    160    INZ(X'00000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000')
D* Ver:3 **
D* Address of MQSCO structure for client connection
D  CNSCP                161    176*   INZ(*NULL)
D* Offset of MQSCO structure for client connection
D  CNSCO                177    180I 0 INZ(0)
D* Ver:4 **
D* Unique Connection Identifier
D  CNCONID              181    204    INZ(X'00000000000000000-
D                                     00000000000000000000000000-
D                                     000000')
D* Offset of MQCSP structure
D  CNSECPO              205    208I 0 INZ(0)
D* Address of MQCSP structure
D  CNSECPP              209    224*   INZ(*NULL)
D* Ver:5 **
D* Address of CCDT URL string
D  CNCCDTUP             225    240*   INZ(*NULL)
D* Offset of CCDT URL string
D  CNCCDTUO             241    244I 0 INZ(0)
```

```
     D* Length of CCDT URL
     D  CNCCDTUL             245    248I 0 INZ(0)
     D* Ver:6 **
     D*
     D*****************************************************************
     D**  End of CMQCNOG                                             **
     D*****************************************************************
```

IBM i MQCSP (parámetros de seguridad) en IBM i

Resumen de la estructura MQCSP para IBM i.

Visión general

Finalidad: la estructura MQCSP permite al servicio de autorización autenticar un ID de usuario y una contraseña. Especifique la estructura de parámetros de seguridad de conexión MQCSP en una llamada MQCONNX.

Conjunto de caracteres y codificación: Los datos de MQCSP deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

- "Campos" en la página 1077
- "Valores iniciales" en la página 1078
- "Declaración RPG" en la página 1079

Campos

La estructura MQCSP contiene los campos siguientes; los campos se describen en **orden alfabético**:

CSAUTHT (entero con signo de 10 dígitos)

Este es el tipo de autenticación que se debe realizar.

Los valores válidos son:

CSAN
No utilice los campos de ID de usuario y contraseña.

CAUIAP
Autentique los campos de ID de usuario y contraseña.

Este es un campo de entrada. El valor inicial de este campo es CSAN.

CSCPPL (entero con signo de 10 dígitos)

Es la longitud de la contraseña que se utilizará en la autenticación.

La longitud máxima de la contraseña no depende de la plataforma. Si la longitud de la contraseña es mayor que la permitida, la solicitud de autenticación falla con un RC2035.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCPPO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes de la contraseña que se va a utilizar en la autenticación.

El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCPPP (puntero)

Es la dirección de la contraseña que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo.

CSCSPUIL (entero con signo de 10 dígitos)

Es la longitud del ID de usuario que se va a utilizar en la autenticación.

La longitud máxima del ID de usuario no depende de la plataforma. Si la longitud del ID de usuario es mayor que la permitida, la solicitud de autenticación falla con un RC2035.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCSPUIO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes del ID de usuario que se va a utilizar en la autenticación.

El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCSPUIP (puntero)

Es la dirección del ID de usuario que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si CSVER es menor que CSVER5.

CSRE1 (serie de caracteres de 4 bytes)

Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSRS2 (serie de caracteres de 8 bytes)

Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

CSSIDV
Identificador de la estructura de parámetros de seguridad.

CSVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

CSVER1
Estructura de parámetros de seguridad de Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

CSVERC
Versión actual de la estructura de parámetros de seguridad.

Siempre es un campo de entrada. El valor inicial de este campo es CSVER1.

Valores iniciales

Tabla 694. Valores iniciales de los campos en MQCNO

Nombre de campo	Nombre de constante	Valor de constante
CSSID	CSSIDV	`'CSP¬'`
CSVER	CSVER1	`1`
CSAUTHT	Ninguna	`0`

Tabla 694. Valores iniciales de los campos en MQCNO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
CSRE1	Ninguna	Nulos
CSCSPUIP	Ninguna	Puntero nulo
CSCSPUIO	Ninguna	0
CSCSPUIL	Ninguna	0
CSRS2	Ninguna	Nulos
CSCPPP	Ninguna	Puntero nulo
CSCPPO	Ninguna	0
CSCPPL	Ninguna	0

Nota:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (estructura de opciones de devolución de llamada de control) en IBM i

Estructura que especifica la función de devolución de llamada de control.

Visión general

Finalidad

La estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control.

La estructura es un parámetro de entrada y salida en la llamada MQCTL .

Versión

La versión actual de MQCTLO es CTLV1.

Juego de caracteres y codificación
Los datos de MQCTLO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

- "Campos" en la página 1080
- "Valores iniciales" en la página 1081
- "Declaración RPG" en la página 1081

Campos

La estructura MQCTLO contiene los campos siguientes; los campos se describen en orden alfabético:

COCONNAREA (entero con signo de 10 dígitos)

Estructura de opciones de control-Campo ConnectionArea .

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo CBCCONNAREA de la estructura MQCBC, que es un parámetro de la llamada MQCB.

Este campo se ignora para todas las operaciones que no sean CTLSR y CTLSW.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

COOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQCTLO.

CTLFQ
Forzar que la llamada MQCTLO falle si el gestor de colas o la conexión está en estado de desactivación temporal.

Especifique GMFIQ, en las opciones MQGMO que se pasan en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

CTLTHR
Esta opción informa al sistema de que la aplicación requiere que todos los consumidores de mensajes, para la misma conexión, se llamen en la misma hebra.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

CTLNO
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. CTLNO está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *COOPT* es CTLNO.

CORSV (entero con signo de 10 dígitos)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco.

COSID (entero con signo de 10 dígitos)

Estructura de opciones de control-Campo StrucId .

Este es el identificador de estructura; el valor debe ser:

CTLSI
Identificador de la estructura de opciones de control.

Siempre es un campo de entrada. El valor inicial de este campo es CTLSI.

COVER (entero con signo de 10 dígitos)

Estructura de opciones de control-Campo Versión.

Este es el número de versión de la estructura; el valor debe ser:

CTLV1
Version-1 Estructura de opciones de control.

La constante siguiente especifica el número de versión de la versión actual:

CTLCV
Versión actual de la estructura de opciones de control.

Siempre es un campo de entrada. El valor inicial de este campo es CTLV1.

Valores iniciales

Tabla 695. Valores iniciales de campos en MQCTLO

Nombre de campo	Nombre de constante	Valor de constante
COSID	CTLSI	'CTLO'
COVER	CTLV1	1
COOPT	CTLNO	Nulos
CORSV	Reservado, campo	
COCONNAREA	Ninguna	Puntero nulo o bytes nulos

Declaración 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 (Cabecera de distribución) en IBM i

La estructura MQDH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión.

Visión general

Finalidad: un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales constan de la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR.

Esta estructura es para uso de aplicaciones especializadas que colocan mensajes directamente en colas de transmisión, o que eliminan mensajes de colas de transmisión (por ejemplo: agentes de canal de mensajes).

Esta estructura no debería ser utilizada por aplicaciones normales que simplemente quieren poner mensajes en listas de distribución. Estas aplicaciones deben utilizar la estructura MQOD para definir los destinos en la lista de distribución, y la estructura MQPMO para especificar propiedades de mensaje o recibir información sobre los mensajes enviados a los destinos individuales.

Conjunto de caracteres y codificación: Los datos de MQDH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **`CodedCharSetId`** y la codificación del gestor de colas local proporcionada por ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación de la MQDH deben establecerse en los campos *`MDCSI`* y *`MDENC`* en:

- El MQMD (si la estructura MQDH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQDH (todos los demás casos).

Uso: Cuando una aplicación coloca un mensaje en una lista de distribución, y algunos o todos los destinos son remotos, el gestor de colas prefija los datos del mensaje de aplicación con las estructuras MQXQH y MQDH, y coloca el mensaje en la cola de transmisión relevante. Por lo tanto, los datos se producen en la secuencia siguiente cuando el mensaje está en una cola de transmisión:

- estructura MQXQH
- Estructura MQDH más matrices de registros MQOR y MQPMR
- Datos de mensaje de aplicación

En función de los destinos, el gestor de colas puede generar más de un mensaje de este tipo y colocarlo en distintas colas de transmisión. En este caso, las estructuras MQDH de esos mensajes identifican distintos subconjuntos de los destinos definidos por la lista de distribución abierta por la aplicación.

Una aplicación que coloca un mensaje de lista de distribución directamente en una cola de transmisión debe ajustarse a la secuencia descrita anteriormente y debe asegurarse de que la estructura MQDH sea correcta. Si la estructura MQDH no es válida, el gestor de colas puede optar por fallar la llamada MQPUT o MQPUT1 con el código de razón RC2135.

Los mensajes se pueden almacenar en una cola en formato de lista de distribución sólo si la cola está definida como que puede dar soporte a los mensajes de lista de distribución (consulte el atributo de cola **`DistLists`** descrito en "Atributos para colas" en la página 1402). Si una aplicación coloca un mensaje de lista de distribución directamente en una cola que no da soporte a listas de distribución, el gestor de colas divide el mensaje de lista de distribución en mensajes individuales y, en su lugar, coloca los mensajes en la cola.

- "Campos" en la página 1082
- "Valores iniciales" en la página 1085
- "Declaración RPG" en la página 1086

Campos

La estructura MQDH contiene los campos siguientes; los campos se describen en **orden alfabético**:

DHCNT (entero con signo de 10 dígitos)

Número de registros MQOR presentes.

Define el número de destinos. Una lista de distribución siempre debe contener al menos un destino, por lo que *`DHCNT`* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

DHCSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen a los registros MQOR y MQPMR.

Especifica el identificador de juego de caracteres de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos de caracteres de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

DHENC (entero con signo de 10 dígitos)

Codificación numérica de los datos que siguen a los registros MQOR y MQPMR.

Especifica la codificación numérica de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos numéricos de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

DHFLG (entero con signo de 10 dígitos)

Distintivos generales.

Se puede especificar el distintivo siguiente:

DHFNEW
Generar nuevos identificadores de mensaje.

Este distintivo indica que se va a generar un nuevo identificador de mensaje para cada destino de la lista de distribución. Esto sólo se puede establecer cuando no hay registros de colocación de mensajes presentes, o cuando los registros están presentes pero no contienen el campo *PRMID* .

El uso de este distintivo difiere la generación de los identificadores de mensaje hasta el último momento posible, es decir, el momento en que el mensaje de lista de distribución se divide finalmente en mensajes individuales. Esto minimiza la cantidad de información de control que debe fluir con el mensaje de lista de distribución.

Cuando una aplicación coloca un mensaje en una lista de distribución, el gestor de colas establece DHFNEW en la MQDH que genera cuando se cumplen las dos sentencias siguientes:

- No hay registros de colocación de mensajes proporcionados por la aplicación, o los registros proporcionados no contienen el campo *PRMID* .
- El campo *MDMID* en MQMD es MINONE, o el campo *PMOPT* en MQPMO incluye PMNMID

Si no se necesitan distintivos, se puede especificar lo siguiente:

DHFNON
Sin distintivos.

Esta constante indica que no se han especificado distintivos. DHFNON está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial de este campo es DHFNON.

DHFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen a los registros MQOR y MQPMR.

Especifica el nombre de formato de los datos que siguen a las matrices de registros MQOD y MQPMR (lo que ocurra en último lugar).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

El valor inicial de este campo es FMNONE.

DHLEN (entero con signo de 10 dígitos)

Longitud de la estructura MQDH más los registros MQOR y MQPMR siguientes.

Es el número de bytes desde el inicio de la estructura MQDH hasta el inicio de los datos de mensaje que siguen a las matrices de registros MQOR y MQPMR. Los datos se producen en la secuencia siguiente:

- estructura MQDH
- Matriz de registros MQOR
- Matriz de registros MQPMR
- Datos de mensaje

Las matrices de registros MQOR y MQPMR se direccionan mediante desplazamientos contenidos en la estructura MQDH. Si estos desplazamientos dan como resultado bytes no utilizados entre una o más de la estructura MQDH, las matrices de registros y los datos de mensaje, dichos bytes no utilizados deben incluirse en el valor de *DHLEN*, pero el gestor de colas no conserva el contenido de dichos bytes. Es válido que la matriz de registros MQPMR preceda a la matriz de registros MQOR.

El valor inicial de este campo es 0.

DHORO (entero con signo de 10 dígitos)

Desplazamiento del primer registro MQOR desde el inicio de MQDH.

Este campo proporciona el desplazamiento en bytes del primer registro de la matriz de registros de objeto MQOR que contiene los nombres de las colas de destino. Hay *DHCNT* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de objeto y el campo anterior) se incluyen en la longitud proporcionada por el campo *DHLEN* .

Una lista de distribución siempre debe contener al menos un destino, por lo que *DHORO* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

DHPRF (entero con signo de 10 dígitos)

Distintivos que indican qué campos MQPMR están presentes.

Se pueden especificar cero o más de los distintivos siguientes:

IDPFM
El campo de identificador de mensaje está presente.

PICD
El campo de identificador de correlación está presente.

IDPFG
El campo identificador de grupo está presente.

PFFB
El campo de comentarios está presente.

PFACC
El campo de señal de contabilidad está presente.

Si no hay campos MQPMR presentes, se puede especificar lo siguiente:

PNONE
No hay campos de registro de colocación de mensaje.

PFNONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

El valor inicial de este campo es PFNONE.

DHPRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro MQPMR desde el inicio de MQDH.

Este campo proporciona el desplazamiento en bytes del primer registro de la matriz de registros de mensajes de colocación MQPMR que contienen las propiedades del mensaje. Si está presente, hay *DHCNT* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de mensaje de colocación y el campo anterior) se incluyen en la longitud proporcionada por el campo *DHLEN* .

Los registros de mensajes de colocación son opcionales; si no se proporcionan registros, *DHPRO* es cero y *DHPRF* tiene el valor PFNONE.

El valor inicial de este campo es 0.

DHSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DHSIDV
Identificador de la estructura de cabecera de distribución.

El valor inicial de este campo es DHSIDV.

DHVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DHVER1
Número de versión para la estructura de cabecera de distribución.

La constante siguiente especifica el número de versión de la versión actual:

DHVERC
Versión actual de la estructura de cabecera de distribución.

El valor inicial de este campo es DHVER1.

Valores iniciales

Tabla 696. Valores iniciales de campos en MQDH

Nombre de campo	Nombre de constante	Valor de constante
DHSID	DHSIDV	'DH¬¬'
DHVER	DHVER1	1
DHLEN	Ninguna	0
DHENC	Ninguna	0
DHCSI	CUNDF	0
DHFMT	FMNONE	Espacios en blanco
DHFLG	DHFNON	0
DHPRF	PNONE	0
DHCNT	Ninguna	0

Tabla 696. Valores iniciales de campos en MQDH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
DHORO	Ninguna	0
DHPRO	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (cabecera Dead-letter) en IBM i

Visión general

Finalidad

La estructura MQDLH describe la información que prefija los datos de mensaje de aplicación de los mensajes en la cola de mensajes no entregados (mensaje no entregado). Un mensaje puede llegar a la cola de mensajes no entregados porque el gestor de colas o el agente de canal de mensajes lo ha redirigido a la cola. Una aplicación puede colocar el mensaje directamente en la cola.

Nombre de formato

FMDLH

Juego de caracteres y codificación

El MQDLH puede estar en el inicio de los datos del mensaje de aplicación. Si es así, los campos de la estructura MQDLH están en el juego de caracteres y la codificación proporcionados por los campos MDCSI y MDENC . Si no es así, el juego de caracteres y la codificación se establecen mediante los campos MDCSI y MDENC en la estructura de cabecera que precede a MQDLH.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Utilización

Las aplicaciones que colocan mensajes directamente en la cola de mensajes no entregados deben prefijar los datos del mensaje con una estructura MQDLH e inicializar los campos con los valores

adecuados. Sin embargo, el gestor de colas no requiere que esté presente una estructura MQDLH o que se especifiquen valores válidos para los campos.

Si un mensaje es demasiado largo para colocarlo en la cola de mensajes no entregados, la aplicación debe considerar la posibilidad de realizar una de las acciones siguientes:

- Trunque los datos del mensaje para que quepan en la cola de mensajes no entregados.
- Anote el mensaje en el almacenamiento auxiliar y coloque un mensaje de informe de excepción en la cola de mensajes no entregados indicando que el mensaje es demasiado largo.
- Descarte el mensaje y devuelva un error a su originador. Si el mensaje es un mensaje crítico. Descarte el mensaje sólo si se sabe que el originador todavía tiene una copia del mensaje. Por ejemplo, un mensaje recibido por un agente de canal de mensajes desde un canal de comunicación.

Cuál de las opciones es apropiada depende del diseño de la aplicación.

El gestor de colas realiza un proceso especial cuando un mensaje que es un segmento se coloca con una estructura MQDLH en la parte frontal. Consulte la descripción de la estructura MQMDE para obtener más detalles.

- "Colocación de mensajes en la cola de mensajes no entregados" en la página 1087
- "Obtención de mensajes de la cola de mensajes no entregados" en la página 1088
- "Campos" en la página 1088
- "Valores iniciales" en la página 1091
- "Declaración RPG" en la página 1092

Colocación de mensajes en la cola de mensajes no entregados

Si se coloca un mensaje en la cola de mensajes no entregados, la estructura MQMD utilizada para la llamada MQPUT o MQPUT1 debe ser idéntica a la MQMD asociada con el mensaje. El MQMD suele ser el que devuelve la llamada MQGET , excepto en los casos siguientes:

- Los campos MDCSI y MDENC deben establecerse en cualquier juego de caracteres y codificación que se utilicen para los campos de la estructura MQDLH .
- El campo MDFMT debe establecerse en FMDLH para indicar que los datos empiezan por una estructura MQDLH .
- Los campos de contexto, MDACC, MDAID, MDAOD, MDPAN, MDPAT, MDPD, MDPTy MDUID deben establecerse utilizando una opción de contexto adecuada a las circunstancias:
  - Una aplicación que coloca en la cola de mensajes no entregados un mensaje que no está relacionado con ningún mensaje anterior debe utilizar la opción PMDEFC . La opción PMDEFC hace que el gestor de colas establezca todos los campos de contexto del descriptor de mensaje en sus valores predeterminados.
  - Una aplicación de servidor que pone en la cola de mensajes no entregados un mensaje que ha recibido debe utilizar la opción PMPASA , para conservar la información de contexto original.
  - Una aplicación de servidor que coloca en la cola de mensajes no entregados una respuesta al mensaje que ha recibido debe utilizar la opción PMPASI . La opción PMPASI conserva la información de identidad pero establece que la información de origen sea la de la aplicación de servidor.
  - Un agente de canal de mensajes que coloca en la cola de mensajes no entregados un mensaje que ha recibido de su canal de comunicación debe utilizar la opción PMSETA . La opción PMSETA conserva la información de contexto original.

En la propia estructura MQDLH , los campos se deben establecer de la forma siguiente:

- Los campos DLCSI, DLENCy *DLFMT* deben establecerse en los valores que describen los datos que siguen a la estructura MQDLH . Estos valores suelen ser los valores del descriptor de mensaje original.
- Los campos de contexto DLPAT, DLPAN, DLPDy DLPT deben establecerse en valores adecuados para la aplicación que coloca el mensaje en la cola de mensajes no entregados. Estos valores no están relacionados con el mensaje original.

- Se deben establecer otros campos según convenga.

La aplicación debe asegurarse de que todos los campos tienen valores válidos y que los campos de caracteres se rellenan con espacios en blanco hasta la longitud definida del campo. Los datos de tipo carácter no deben terminarse prematuramente utilizando un carácter nulo. El gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQDLH .

Obtención de mensajes de la cola de mensajes no entregados

Las aplicaciones que obtienen mensajes de la cola de mensajes no entregados deben verificar que los mensajes empiezan con una estructura MQDLH . La aplicación puede determinar si existe una estructura MQDLH examinando el campo MDFMT en el descriptor de mensaje MQMD. Si el campo tiene el valor FMDLH, los datos del mensaje empiezan por una estructura MQDLH . Es posible que los mensajes de la cola de mensajes no entregados se trunquen si originalmente eran demasiado largos para la cola para la que estaban pensados.

Campos

La estructura MQDLH contiene los campos siguientes; los campos se describen en orden alfabético:

DLCSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen a MQDLH.

DLCSI especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQDLH . Los datos suelen ser del mensaje original. No se aplica a los datos de tipo carácter de la propia estructura MQDLH .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres de los datos que siguen a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT .

CSINHT no se puede utilizar si el valor del campo MDPAT en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

DLDM (serie de caracteres de 48 bytes)

Nombre del gestor de colas de destino original.

Es el nombre del gestor de colas que era el destino original del mensaje.

La longitud de este campo la proporciona LNQMN. El valor inicial de este campo es de 48 caracteres en blanco.

DLDQ (serie de caracteres de 48 bytes)

Nombre de la cola de destino original.

Es el nombre de la cola de mensajes que era el destino original del mensaje.

La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

DLENC (entero con signo de 10 dígitos)

Codificación numérica de datos que sigue a MQDLH.

DLENC especifica la codificación numérica de los datos que siguen a la estructura MQDLH . Los datos suelen ser del mensaje original. No se aplica a los datos numéricos de la propia estructura MQDLH .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

DLFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen a MQDLH.

Especifica el nombre de formato de los datos que siguen a la estructura MQDLH (normalmente los datos del mensaje original).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para el campo MDFMT en MQMD.

La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

DLPAN (serie de caracteres de 28 bytes)

Nombre de la aplicación que ha colocado el mensaje en la cola de mensajes no entregados.

El formato del nombre depende del campo DLPAT . Consulte la descripción del campo MDPAN en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133.

Si es el gestor de colas el que redirige el mensaje a la cola de mensajes no entregados, DLPAN contiene los primeros 28 caracteres del nombre del gestor de colas. El nombre se rellena con espacios en blanco si es necesario.

La longitud de este campo la proporciona LNPAN. El valor inicial de este campo es de 28 caracteres en blanco.

DLPAT (entero con signo de 10 dígitos)

Tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados.

Este campo tiene el mismo significado que el campo MDPAT en el descriptor de mensaje MQMD (consulte "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener más detalles).

Si es el gestor de colas el que redirige el mensaje a la cola de mensajes no entregados, DLPAT tiene el valor ATQM.

El valor inicial de este campo es 0.

DLPD (serie de caracteres de 8 bytes)

Fecha en la que se colocó el mensaje en la cola de mensajes no entregados (mensaje no entregado).

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- YYYYMMDD

donde los caracteres representan:

YYYY
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos DLPD y DLPT , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona LNPDAT. El valor inicial de este campo es de ocho caracteres en blanco.

DLPT (serie de caracteres de 8 bytes)

Hora a la que se ha colocado el mensaje en la cola de mensajes no entregados.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan (en orden):

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte la nota más adelante en este tema)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de hora preciso, es posible que se devuelvan 60 o 61 durante los segundos en DLPT. El segundo adicional se produce cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos DLPD y DLPT , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona LNPTIM. El valor inicial de este campo es de ocho caracteres en blanco.

DLREA (entero con signo de 10 dígitos)

El mensaje de razón ha llegado a la cola de mensajes no entregados (mensaje no entregado).

Esto identifica la razón por la que el mensaje se ha colocado en la cola de mensajes no entregados en lugar de en la cola de destino original. Debe ser uno de los valores FB* o RC* (por ejemplo, RC2053). Consulte la descripción del campo *MDFB* en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener detalles de los valores FB* comunes que se pueden producir.

Si el valor está en el rango de FBIFST a FBILST, el código de error IMS real se puede determinar restando FBIERR del valor del campo *DLREA* .

Algunos valores de FB* sólo aparecen en este campo. Están relacionados con mensajes de repositorio, mensajes desencadenantes o mensajes de cola de transmisión que se transfieren a la cola de mensajes no entregados. Estos valores son:

FBABEG
No se puede iniciar la aplicación.

Una aplicación que procesa un mensaje desencadenante no ha podido iniciar la aplicación especificada en el campo TMAI del mensaje desencadenante; consulte "MQTM-Mensaje de desencadenante" en la página 1264.

FBATYP
Error de tipo de aplicación.

Una aplicación que procesa un mensaje desencadenante no ha podido iniciar la aplicación porque el campo TMAT del mensaje desencadenante no es válido; consulte "MQTM-Mensaje de desencadenante" en la página 1264.

FBBOCD
Se ha suprimido el canal de clúster receptor.

El mensaje estaba en una cola de transmisión de clúster pensada para una cola de clúster que se ha abierto con la opción FBIERR . El canal de clúster receptor remoto que se va a utilizar

para transmitir el mensaje a la cola de destino se ha suprimido antes de que se pudiera enviar el mensaje. Puesto que se ha especificado FBIERR , sólo se puede utilizar el canal seleccionado al abrir la cola para transmitir el mensaje. Como este canal ya no está disponible, el mensaje se ha colocado en la cola de mensajes no entregados.

FBNARM
El mensaje no es un mensaje de repositorio.

FBSBCX
Mensaje detenido por salida de definición automática de canal.

FBSBMX
Mensaje detenido por salida de mensajes de canal.

FBTM
La estructura MQTM no es válida o falta.

El campo MDFMT en MQMD especifica FMTM, pero el mensaje no empieza por una estructura MQTM válida. Por ejemplo, es posible que el captador de atención mnemotécnico de *TMSID* no sea válido. Es posible que *TMVER* no se reconozca. La longitud del mensaje desencadenante puede ser insuficiente para contener la estructura MQTM .

FBXQME
El mensaje en la cola de transmisión no está en el formato correcto.

Un agente de canal de mensajes ha detectado que un mensaje de la cola de transmisión no tiene el formato correcto. El agente de canal de mensajes coloca el mensaje en la cola de mensajes no entregados utilizando este código de retorno.

El valor inicial de este campo es RCNONE.

DLSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DLSIDV
Identificador de la estructura de cabecera de mensaje no entregado.

El valor inicial de este campo es DLSIDV.

DLVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DLVER1
Número de versión para la estructura de cabecera de mensaje no entregado.

La constante siguiente especifica el número de versión de la versión actual:

DLVERC
Versión actual de la estructura de cabecera de mensaje no entregado.

El valor inicial de este campo es DLVER1.

Valores iniciales

Tabla 697. Valores iniciales de los campos en MQDLH

Nombre de campo	Nombre de constante	Valor de constante
IDDLS	DLSIDV	'DLH¬'
DLVER	DLVER1	1
DLREA	RCNONE	0

Tabla 697. Valores iniciales de los campos en MQDLH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
DLDQ	Ninguna	Espacios en blanco
DLDM	Ninguna	Espacios en blanco
DLENC	Ninguna	0
DLCSI	CSUNDF	0
DLFMT	FMNONE	Espacios en blanco
DLPAT	Ninguna	0
DLPAN	Ninguna	Espacios en blanco
DLPD	Ninguna	Espacios en blanco
DLPT	Ninguna	Espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Suprimir opciones de manejador de mensajes) en IBM i

La estructura **MQDMHO** permite a las aplicaciones especificar opciones que controlan cómo se suprimen los manejadores de mensajes.

Visión general

Finalidad: la estructura es un parámetro de entrada en la llamada **MQDLTMH** .

Conjunto de caracteres y codificación: los datos de **MQDMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1093
- “Valores iniciales” en la página 1093
- “Declaración RPG” en la página 1093

Campos

La estructura MQDMHO contiene los campos siguientes; los campos se describen en **orden alfabético**:

DMOPT (entero con signo de 10 dígitos)

El valor debe ser:

DMNONE

No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es **DMNONE**.

DMSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

DMSIDV

Identificador de la estructura de opciones de manejador de mensajes de supresión.

Siempre es un campo de entrada. El valor inicial de este campo es **DMSIDV**.

DMVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

DMVER1
Version-1 suprime la estructura de opciones del manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

DMVERC
Versión actual de la estructura de opciones de manejador de mensajes de supresión.

Siempre es un campo de entrada. El valor inicial de este campo es **DMVER1**.

Valores iniciales

Tabla 698. Valores iniciales de los campos en MQDMHO

Nombre de campo	Nombre de constante	Valor de constante
DMSID	DMSIDV	'DMHO'
DMVER	DMVER1	1
DMOPT	DMNONE	0

Declaración 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 (Suprimir opciones de propiedad de mensaje) en IBM i

Estructura que define las opciones de supresión de propiedad de mensaje.

Visión general

Finalidad: la estructura MQDMPO permite a las aplicaciones especificar opciones que controlan cómo se suprimen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Conjunto de caracteres y codificación: los datos de MQDMPO deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- "Campos" en la página 1094
- "Valores iniciales" en la página 1095
- "Declaración RPG" en la página 1095

Campos

La estructura MQDMPO contiene los campos siguientes; los campos se describen en orden alfabético:

DPOPT (entero con signo de 10 dígitos)

Suprimir estructura de opciones de propiedad de mensaje-Campo DPOPT.

Opciones de ubicación: Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de la propiedad.

PDELF
Suprime la primera propiedad que coincide con el nombre especificado.

DPDELC
Suprime la propiedad a la que apunta el cursor de propiedad; es decir, la propiedad que se ha consultado por última vez utilizando la opción IPINQF o la opción IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje. También se restablece cuando se especifica el descriptor de mensaje en el campo *HMSG* de MQGMO en una llamada MQGET o estructura MQPMO en una llamada MQPUT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, o cuando se especifica el descriptor de mensaje en el campo *HMSG* de la estructura MQGMO en una estructura MQGET en una llamada MQGET o estructura MQPMO en una llamada MQPUT.

La llamada falla con el código de terminación CCFAIL y la razón RC2471 si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad. También falla con estos códigos si la propiedad a la que apunta el cursor de propiedad ya se ha suprimido.

Si no se requiere ninguna de estas opciones, se puede utilizar la siguiente opción:

DPNONE
No se ha especificado ninguna opción.

El valor inicial de este campo de entrada es DPDELF.

DPSID (entero con signo de 10 dígitos)

Suprimir estructura de opciones de propiedad de mensaje-campo DPSID.

Es el identificador de estructura. El valor debe ser:

DPSID
Identificador para la estructura de opciones de propiedad de mensaje de supresión.

Este campo siempre es un campo de entrada. El valor inicial de este campo es DPSIDV.

DPVER (entero con signo de 10 dígitos)

Suprimir estructura de opciones de propiedad de mensaje-Campo DPVER.

Es el número de versión de la estructura. El valor debe ser:

DPVER1
Número de versión para la estructura de opciones de suprimir propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

DPVERC
Versión actual de la estructura de opciones de supresión de propiedades de mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es DPVER1.

Valores iniciales

Tabla 699. Valores iniciales de campos en MQDPMO

Nombre de campo	Nombre de constante	Valor de constante
DPSID	DPSID	'DMPO'
DPVER	DPVER1	1
DPOPT	Opciones que controlan la acción de MQDLTMP	DPNONE

Declaración 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 (cabecera PCF incorporada) en IBM i

Visión general

Finalidad
La estructura MQEPH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de formato de mandato programable (PCF). El campo *EPPFH* define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otras cabeceras.

Nombre de formato
PFMT

Juego de caracteres y codificación
Los datos de MQEPH deben estar en el juego de caracteres y la codificación del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CCSID** .

Establezca el juego de caracteres y la codificación de MQEPH en los campos *MDCSI* y *MDENC* en:

- El MQMD (si la estructura MQEPH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQEPH (todos los demás casos).

Utilización

No puede utilizar estructuras MQEPH para enviar mandatos al servidor de mandatos o a cualquier otro servidor que acepte PCF del gestor de colas.

De forma similar, el servidor de mandatos o cualquier otro servidor de aceptación de PCF de gestor de colas no generan respuestas o sucesos que contengan estructuras MQEPH.

- "Campos" en la página 1096
- "Valores iniciales" en la página 1097
- "Declaración RPG" en la página 1098

Campos

La estructura MQEPH contiene los campos siguientes; los campos se describen en **orden alfabético**:

EPCSI (entero con signo de 10 dígitos)

Es el identificador de juego de caracteres de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es EPCUND.

EPENC (entero con signo de 10 dígitos)

Es la codificación numérica de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es 0.

EPFLG (entero con signo de 10 dígitos)

Están disponibles los valores siguientes:

EPNONE
No se han especificado distintivos. *MDCSI* EPNONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

EPCSEM
El juego de caracteres de los parámetros que contienen datos de caracteres se especifica individualmente dentro del campo *CCSID* en cada estructura. El juego de caracteres de los campos *EPSID* y *EPFMT* están definidos por *CCSID* en la estructura de cabecera que precede a la estructura MQEPH, o por el campo *MDCSI* en el MQMD si el MQEPH está al principio del mensaje.

El valor inicial de este campo es EPNONE.

EPFMT (serie de caracteres de 8 bytes)

Es el nombre de formato de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados.

El valor inicial de este campo es EPFMNO.

EPLEN (entero con signo de 10 dígitos)

Es la cantidad de datos que preceden a la siguiente estructura de cabecera. Esto incluye:

- Longitud de la cabecera MQEPH
- La longitud de todos los parámetros PCF que siguen a la cabecera
- Cualquier relleno en blanco que siga a estos parámetros

EPLEN debe ser un múltiplo de 4.

La parte de longitud fija de la estructura está definida por EPSTLF.

El valor inicial de este campo es 68.

EPPCFH (MQCFH)

Es la cabecera de formato de mandato programable (PCF), que define los parámetros PCF que siguen a la estructura MQEPH. Esto le permite seguir los datos del mensaje PCF con otras cabeceras.

La cabecera PCF se define inicialmente con los valores siguientes:

Tabla 700. Valores iniciales de los campos en EPPCFH

Nombre de campo	Nombre de constante	Valor de constante
EP3TYP	CFTNON	0
EP3LEN	FHLENV	36
EP3VER	FHVER3	3
EP3CMD	CMNONE	0
EP3SEQ	Ninguna	1
EP3CTL	CFCLST	1
EEP3CC	CCOK	0
EP3REA	RCNONE	0
EP3CNT	Ninguna	0

La aplicación debe cambiar EP3TYP de CFTNON a un tipo de estructura válido para el uso que está haciendo de la cabecera PCF incorporada.

EPSID (serie de caracteres de 4 bytes)

El valor debe ser:

EPSTID
Identificador de la estructura de cabecera PCF incorporada.

El valor inicial de este campo es EPSTID.

EPVER (entero con signo de 10 dígitos)

El valor puede ser:

EPVER1
Número de versión para la estructura de cabecera PCF incorporada.

La constante siguiente especifica el número de versión de la versión actual:

EPVER3
Versión actual de la estructura de cabecera PCF incorporada.

El valor inicial de este campo es EPVER3.

Valores iniciales

Tabla 701. Valores iniciales de los campos en MQEPH

Nombre de campo	Nombre de constante	Valor de constante
EPSID	EPSTID	'EP¬¬'
EPVER	EPVER1	1
EPLEN	ESTLF	68

Tabla 701. Valores iniciales de los campos en MQEPH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
EPENC	Ninguna	0
EPCSI	EPCUND	0
EPFMT	EPFMNO	Espacios en blanco
EPFLG	EPNONE	0
EPPCFH	Nombres y valores tal como se definen en Tabla 700 en la página 1097	0

Nota:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Opciones de obtención de mensajes) en IBM i

La estructura MQGMO permite a la aplicación especificar opciones que controlan cómo se eliminan los mensajes de las colas.

Visión general

Finalidad

La estructura es un parámetro de entrada/salida en la llamada MQGET.

Versión

La versión actual de MQGMO es GMVER4. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQGMO soportada por el entorno, pero con el valor inicial del campo *GMVER* establecido en GMVER1. Para utilizar campos que no están

presentes en la estructura version-1 , la aplicación debe establecer el campo *GMVER* en el número de versión de la versión necesaria.

Juego de caracteres y codificación
Los datos de MQGMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

- "Campos" en la página 1099
- "Valores iniciales" en la página 1120
- "Declaración RPG" en la página 1120

Campos

La estructura MQGMO contiene los campos siguientes; los campos se describen en orden alfabético:

GMGST (serie de caracteres de 1 byte)

Distintivo que indica si el mensaje recuperado está en un grupo.

Tiene uno de los siguientes valores:

GSNIG
El mensaje no está en un grupo.

GSMIG
El mensaje está en un grupo, pero no es el último del grupo.

GSLMIG
El mensaje es el último del grupo.

Este valor también es el valor devuelto si el grupo consta de un solo mensaje.

Este campo es un campo de salida. El valor inicial de este campo es GSNIG. Este campo se ignora si *GMVER* es menor que GMVER2.

GMMH (entero con signo de 10 dígitos)

descriptor de contexto de mensaje

Si se especifica la opción GMPRAQ y el atributo de cola PRPCTL no está establecido en PRPRFH, este es el handle de un mensaje que se llena con las propiedades del mensaje que se está recuperando de la cola. El descriptor de contexto se crea mediante una llamada MQCRTMH. Las propiedades que ya están asociadas al descriptor se borran antes de recuperar un mensaje.

También se puede especificar el valor siguiente:
MQHM_NONE
No se ha proporcionado ningún manejador de mensajes.

No es necesario ningún descriptor de mensaje en la llamada MQGET si se proporciona un descriptor de mensaje válido y se utiliza en la salida para contener las propiedades del mensaje, el descriptor de mensaje asociado con el descriptor de mensaje se utiliza para los campos de entrada.

Si se especifica un descriptor de mensaje en la llamada MQGET, siempre tiene prioridad sobre el descriptor de mensaje asociado a un descriptor de mensaje.

Si se especifica GMPRRF, o se especifica GMPRAQ y el atributo de cola PRPCTL es PRPRFH, la llamada falla con el código de razón RC2026 cuando no se especifica ningún parámetro de descriptor de mensaje.

Al volver de la llamada MQGET, las propiedades y el descriptor de mensaje asociados con este descriptor de mensaje se actualizan para reflejar el estado del mensaje recuperado (así como el descriptor de mensaje si se ha proporcionado uno en la llamada MQGET). A continuación, se pueden consultar las propiedades del mensaje utilizando la llamada MQINQMP.

Excepto para las extensiones de descriptor de mensaje, cuando están presentes, una propiedad que se puede consultar con la llamada MQINQMP no está contenida en los datos del mensaje; si el mensaje de la cola contenía propiedades en los datos del mensaje, estos se eliminan de los datos del mensaje antes de que se devuelvan los datos a la aplicación.

Si no se proporciona ningún descriptor de mensaje o la versión es inferior a GMVER4 , debe proporcionar un descriptor de mensaje válido en la llamada MQGET. Cualquier propiedad de mensaje (excepto las propiedades contenidas en el descriptor de mensaje) se devuelve en los datos de mensaje sujetos al valor de las opciones de propiedad en la estructura MQGMO y el atributo de cola PRPCTL.

Este campo es siempre un campo de entrada. El valor inicial de este campo es HMNONE. Este campo se ignora si *GMVER* es menor que GMVER4.

GMMO (entero con signo de 10 dígitos)

Opciones que controlan los criterios de selección utilizados para MQGET.

Estas opciones permiten a la aplicación elegir qué campos del parámetro **MSGDSC** se utilizan para seleccionar el mensaje devuelto por la llamada MQGET. La aplicación establece las opciones necesarias en este campo y, a continuación, establece los campos correspondientes en el parámetro **MSGDSC** en los valores necesarios para esos campos. Sólo los mensajes que tienen estos valores en MQMD para el mensaje son candidatos para la recuperación utilizando ese parámetro **MSGDSC** en la llamada MQGET. Los campos para los que no se ha especificado la opción de coincidencia correspondiente se ignoran al seleccionar el mensaje que se devolverá. Si no se va a utilizar ningún criterio de selección en la llamada MQGET (es decir, cualquier mensaje es aceptable), *GMMO* debe establecerse en MONONE.

Si se especifica GMLOGO, sólo determinados mensajes pueden ser devueltos por la siguiente llamada MQGET:

- Si no hay ningún grupo actual o mensaje lógico, sólo se podrán devolver los mensajes que tengan *MDSEQ* igual a 1 y *MDOFF* igual a 0 . En esta situación, se pueden utilizar una o varias de las opciones siguientes para seleccionar cuál de los mensajes elegibles es el que se devuelve:
  - MOMSGI
  - MOCORI
  - MOGRPI
- Si hay un grupo o mensaje lógico actual, sólo el siguiente mensaje del grupo o segmento siguiente del mensaje lógico es elegible para su devolución, y esto no se puede modificar especificando las opciones MO*.

En ambos casos, se pueden especificar opciones de coincidencia que no son aplicables, pero el valor del campo relevante en el parámetro **MSGDSC** debe coincidir con el valor del campo correspondiente en el mensaje que se va a devolver; la llamada falla con el código de razón RC2247 si no se cumple esta condición.

GMMO se ignora si se especifica GMMUC o GMBRWC.

Se pueden especificar una o más de las opciones siguientes:

MOMSGI

Recuperar mensaje con el identificador de mensaje especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de mensaje que coincida con el valor del campo *MDMID* en el parámetro **MSGDSC** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si no se especifica esta opción, el campo *MDMID* del parámetro **MSGDSC** se ignora y cualquier identificador de mensaje coincide.

Nota: El identificador de mensaje MINONE es un valor especial que coincide con cualquier identificador de mensaje del MQMD del mensaje. Por lo tanto, especificar MOMSGI con MINONE es lo mismo que no especificar MOMSGI.

MOCORI
Recuperar mensaje con el identificador de correlación especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de correlación que coincida con el valor del campo *MDCID* en el parámetro **MSGDSC** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de mensaje).

Si no se especifica esta opción, el campo *MDCID* del parámetro **MSGDSC** se ignora y cualquier identificador de correlación coincide.

Nota: El identificador de correlación CINONE es un valor especial que coincide con cualquier identificador de correlación del MQMD del mensaje. Por lo tanto, especificar MOCORI con CINONE es lo mismo que no especificar MOCORI.

MOGRPI
Recuperar mensaje con el identificador de grupo especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de grupo que coincida con el valor del campo *MDGID* en el parámetro **MSGDSC** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si no se especifica esta opción, el campo *MDGID* del parámetro **MSGDSC** se ignora y cualquier identificador de grupo coincide.

Nota: El identificador de grupo GINONE es un valor especial que coincide con cualquier identificador de grupo del MQMD del mensaje. Por lo tanto, especificar MOGRPI con GINONE es lo mismo que no especificar MOGRPI.

MOSEQN
Recuperar mensaje con el número de secuencia de mensaje especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un número de secuencia de mensaje que coincida con el valor del campo *MDSEQ* en el parámetro **MSGDSC** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de grupo).

Si no se especifica esta opción, el campo *MDSEQ* del parámetro **MSGDSC** se ignora y cualquier número de secuencia de mensaje coincide.

MOOFFS
Recuperar mensaje con desplazamiento especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un desplazamiento que coincida con el valor del campo *MDOFF* en el parámetro **MSGDSC** de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que se pueda aplicar (por ejemplo, el número de secuencia de mensaje).

Si no se especifica esta opción, el campo *MDOFF* del parámetro **MSGDSC** se ignora y cualquier desplazamiento coincide.

Si no se especifica ninguna de las opciones descritas, se puede utilizar la siguiente opción:

MONONA
No hay coincidencias.

Esta opción especifica que no se deben utilizar coincidencias al seleccionar el mensaje que se va a devolver; por lo tanto, todos los mensajes de la cola son elegibles para la recuperación (pero están sujetos al control de las opciones GMAMSA, GMASGA y GMCMPM).

MONONE está definido para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción MO*, pero como su valor es cero, no se puede detectar dicho uso.

Este campo es un campo de entrada. El valor inicial de este campo es MOMSGI con MOCORI. Este campo se ignora si *GMVER* es menor que GMVER2.

Nota: El valor inicial del campo *GMMO* se define para la compatibilidad con gestores de colas de versiones anteriores. Sin embargo, al leer una serie de mensajes de una cola sin utilizar criterios de selección, este valor inicial requiere que la aplicación restablezca los campos *MDMID* y *MDCID* en MINONE y CINONE antes de cada llamada MQGET. La necesidad de restablecer *MDMID* y *MDCID* se puede evitar estableciendo *GMVER* en GMVER2y *GMMO* en MONONE.

GMOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQGET.

Se pueden especificar cero o más de las siguientes opciones descritas. Si se necesita más de uno, se pueden añadir los valores (no añadir la misma constante más de una vez). Las combinaciones de opciones que no son válidas se anotan; todas las demás combinaciones son válidas.

Opciones de espera: Las opciones siguientes están relacionadas con la espera de que lleguen mensajes a la cola:

GMWT
Espere a que llegue el mensaje.

La aplicación debe esperar hasta que llegue un mensaje adecuado. El tiempo máximo que la aplicación espera se especifica en *GMWI*.

Si se inhiben las solicitudes MQGET o las solicitudes MQGET se inhiben mientras se espera, la espera se cancela y la llamada se completa con CCFAIL y el código de razón RC2016, independientemente de si hay mensajes adecuados en la cola.

Esta opción se puede utilizar con las opciones GMBRWF o GMBRWN.

Si varias aplicaciones están esperando en la misma cola compartida, la aplicación o las aplicaciones que se activan cuando llega un mensaje adecuado se describen más adelante en esta sección.

Nota: En la descripción siguiente, una llamada MQGET para examinar es aquella que especifica una de las opciones de examinar, pero no GMLK; una llamada MQGET que especifica la opción GMLK se trata como una llamada no examinar.

- Si una o más llamadas MQGET no examinadas están en espera, pero ninguna llamada MQGET examinada está en espera, se activa una.
- Si una o más llamadas MQGET de examen está en espera, pero ninguna llamada MQGET de no examen está en espera, todas están activadas.
- Si una o más llamadas MQGET no examinadas y una o más llamadas MQGET examinadas están en espera, se activa una llamada MQGET no examinada y ninguna, alguna o todas las llamadas MQGET examinadas. (El número de llamadas MQGET de examen activadas no se puede predecir, porque depende de las consideraciones de planificación del sistema operativo y de otros factores.)

Si hay más de una llamada MQGET sin examinar en espera en la misma cola, sólo se activa una; en esta situación, el gestor de colas intenta dar prioridad a las llamadas sin examinar en espera en el orden siguiente:

1. Solicitudes get-wait específicas que sólo pueden satisfacerse mediante determinados mensajes, por ejemplo, aquellos con un *MDMID* o *MDCID* específico (o ambos).
2. Solicitudes de obtención de espera generales que pueden ser satisfechas por cualquier mensaje.

Deben tenerse en cuenta los puntos siguientes:

- Dentro de la primera categoría, no se otorga ninguna prioridad adicional a las solicitudes get-wait más específicas, por ejemplo, las que especifican *MDMID* y *MDCID*.
- Dentro de cualquiera de las dos categorías, no se puede predecir qué aplicación está seleccionada. En concreto, la aplicación que más espera no es necesariamente la seleccionada.
- La longitud de vía de acceso y las consideraciones de planificación de prioridad del sistema operativo pueden significar que una aplicación en espera de una prioridad de sistema operativo inferior a la esperada recupera el mensaje.
- También puede suceder que una aplicación que no está en espera recupere el mensaje en lugar de uno que sí lo está.

GMWT se ignora si se especifica con GMBRWC o GMMUC; no se genera ningún error.

GMNWT
Devolver inmediatamente si no hay ningún mensaje adecuado.

La aplicación no debe esperar si no hay ningún mensaje adecuado disponible. Esto es lo contrario de la opción GMWT, y se define para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

GMFIQ
Fallar si el gestor de colas se está desactivando temporalmente.

Esta opción fuerza a que la llamada MQGET falle si el gestor de colas está en estado de desactivación temporal.

Si esta opción se especifica junto con GMWT, y la espera está pendiente en el momento en que el gestor de colas entra en el estado de desactivación temporal:

- La espera se cancela y la llamada devuelve el código de terminación CCFAIL con el código de razón RC2161 .

Si no se especifica GMFIQ y el gestor de colas entra en estado de desactivación temporal, la espera no se cancela.

Opciones de punto de sincronismo: Las opciones siguientes están relacionadas con la participación de la llamada MQGET dentro de una unidad de trabajo:

GMSYP
Obtener mensaje con control de punto de sincronismo.

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

Si no se especifica esta opción o GMNSYP, la petición de obtención no está dentro de una unidad de trabajo.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWF
- GMBRWC
- GMBRWN
- GMLK
- GMNSYP
- GMPSYP
- GMUNLK

GMPSYP
Obtener mensaje con control de punto de sincronismo si el mensaje es persistente.

La solicitud es operar dentro de los protocolos de unidad de trabajo normales, pero sólo si el mensaje recuperado es persistente. Un mensaje persistente tiene el valor PEPER en el campo *MDPER* en MQMD.

- Si el mensaje es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado GMSYP.
- Si el mensaje no es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado GMNSYP (consulte la sección siguiente para obtener más detalles).

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWF
- GMBRWC
- GMBRWN
- GMCMPM
- GMNSYP
- GMSYP
- GMUNLK

GMNSYP

Obtener mensaje sin control de punto de sincronismo.

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje se suprime de la cola inmediatamente (a menos que se trate de una solicitud de examen). El mensaje no puede volver a estar disponible restituir la unidad de trabajo.

Esta opción se asume si se especifica GMBRWF o GMBRWN.

Si no se especifica esta opción y GMSYP, la solicitud de obtención no está dentro de una unidad de trabajo.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMSYP
- GMPSYP

Opciones de examen: Las opciones siguientes están relacionadas con el examen de mensajes en la cola:

GMBRWF

Examinar desde el inicio de la cola.

Cuando se abre una cola con la opción OOBRW, se establece un cursor para examinar, situado lógicamente antes del primer mensaje de la cola. Las llamadas MQGET posteriores que especifiquen la opción GMBRWF, GMBRWN o GMBRWC pueden utilizarse para recuperar mensajes de la cola de forma no destructiva. El cursor para examinar marca la posición, dentro de los mensajes de la cola, desde la que la siguiente llamada MQGET con GMBRWN busca un mensaje adecuado.

Una llamada MQGET con GMBRWF hace que se ignore la posición anterior del cursor para examinar. Se recupera el primer mensaje de la cola que cumple las condiciones especificadas en el descriptor de mensaje. El mensaje permanece en la cola y el cursor para examinar se coloca en este mensaje.

Después de esta llamada, el cursor para examinar se coloca en el mensaje que se ha devuelto. Si el mensaje se elimina de la cola antes de que se emita la siguiente llamada MQGET con GMBRWN, el cursor para examinar permanece en la posición de la cola que ocupaba el mensaje, aunque esa posición esté ahora vacía.

A continuación, la opción GMMUC se puede utilizar con una llamada MQGET sin examinar si es necesario, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET para no examinar utilizando el mismo descriptor de contexto *HOBJ* . Tampoco se mueve mediante una llamada MQGET para examinar que devuelve un código de terminación de CCFAIL, o un código de razón de RC2080 .

La opción GMLK se puede especificar junto con esta opción, para hacer que el mensaje que se examina se bloquee.

GMBRWF puede especificarse con cualquier combinación válida de las opciones GM* y MO* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica GMLOGO, los mensajes se examinan en orden lógico. Si se omite esta opción, los mensajes se examinan en orden físico. Cuando se especifica GMBRWF, es posible conmutar entre el orden lógico y el orden físico, pero las llamadas MQGET posteriores que utilizan GMBRWN deben examinar la cola en el mismo orden que la llamada más reciente que ha especificado GMBRWF para el descriptor de contexto de cola.

La información de grupo y segmento que el gestor de colas retiene para las llamadas MQGET que examinan mensajes en la cola, está separada de la información de grupo y segmento que el gestor de colas retiene para las llamadas MQGET que eliminan mensajes de la cola. Cuando se especifica GMBRWF, el gestor de colas ignora la información de grupo y segmento para examinar y explora la cola como si no hubiera ningún grupo actual y ningún mensaje lógico actual. Si la llamada MQGET es satisfactoria (código de terminación CCOK o CCWARN), la información de grupo y segmento para examinar se establece en la del mensaje devuelto; si la llamada falla, la información de grupo y segmento permanece igual que antes de la llamada.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se ha abierto para examinar.

GMBRWN

Examinar desde la posición actual en cola.

El cursor para examinar se avanza al siguiente mensaje de la cola que cumple los criterios de selección especificados en la llamada MQGET. El mensaje se devuelve a la aplicación, pero permanece en la cola.

Después de que se haya abierto una cola para examinar, la primera llamada de examinar que utiliza el descriptor de contexto tiene el mismo efecto si especifica la opción GMBRWF o GMBRWN.

Si el mensaje se elimina de la cola antes de que se emita la siguiente llamada MQGET con GMBRWN, el cursor para examinar permanece lógicamente en la posición de la cola que ocupaba el mensaje, aunque esa posición esté ahora vacía.

Los mensajes se almacenan en la cola de una de estas dos maneras:

- FIFO dentro de prioridad (MSPRIO), o
- FIFO independientemente de la prioridad (MSFIFO)

El atributo de cola **MsgDeliverySequence** indica qué método se aplica (consulte “Atributos para colas” en la página 1402 para obtener más detalles).

Si la cola tiene un *MsgDeliverySequence* de MSPRIO y llega un mensaje a la cola que tiene una prioridad más alta que la a la que apunta actualmente el cursor para examinar, ese mensaje no se encuentra durante el barrido actual de la cola utilizando GMBRWN. Sólo se puede encontrar

después de que el cursor para examinar se haya restablecido con GMBRWF (o volviendo a abrir la cola).

La opción GMMUC se puede utilizar posteriormente con una llamada MQGET sin examinar si es necesario, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante llamadas MQGET para no examinar utilizando el mismo descriptor de contexto *HOBJ* .

La opción GMLK se puede especificar junto con esta opción, para hacer que el mensaje que se examina se bloquee.

GMBRWN puede especificarse con cualquier combinación válida de las opciones GM* y MO* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica GMLOGO, los mensajes se examinan en orden lógico. Si se omite esta opción, los mensajes se examinan en orden físico. Cuando se especifica GMBRWF, es posible conmutar entre el orden lógico y el orden físico, pero las llamadas MQGET posteriores que utilizan GMBRWN deben examinar la cola en el mismo orden que la llamada más reciente que ha especificado GMBRWF para el descriptor de contexto de cola. La llamada falla con el código de razón RC2259 si no se cumple esta condición.

Nota: Es necesario tener especial cuidado si se utiliza una llamada MQGET para examinar más allá del final de un grupo de mensajes (o mensaje lógico que no está en un grupo) cuando no se especifica GMLOGO. Por ejemplo, si el último mensaje del grupo precede al primer mensaje del grupo en la cola, el uso de GMBRWN para examinar más allá del final del grupo, especificando MOSEQN con *MDSEQ* establecido en 1 (para encontrar el primer mensaje del siguiente grupo) devolvería de nuevo el primer mensaje del grupo ya examinado. Esto podría ocurrir de forma inmediata, o después de varias llamadas MQGET (si hay grupos intermedios).

La posibilidad de un bucle infinito se puede evitar abriendo la cola dos veces para examinar:

- Utilice el primer descriptor para examinar solamente el primer mensaje de cada grupo.
- Utilice el segundo descriptor para examinar solamente los mensajes de un grupo específico.
- Utilice las opciones MO* para mover el segundo cursor de examen a la posición del primer cursor de examen, antes de examinar los mensajes del grupo.
- No utilice GMBRWN para examinar más allá del final de un grupo.

La información de grupo y segmento que el gestor de colas retiene para las llamadas MQGET que examinan mensajes en la cola, está separada de la información de grupo y segmento que retiene para las llamadas MQGET que eliminan mensajes de la cola.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWF
- GMBRWC
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se ha abierto para examinar.

GMBRWC

Examine el mensaje bajo el cursor para examinar.

Esta opción hace que el mensaje al que apunta el cursor para examinar se recupere de forma no destructiva, independientemente de las opciones MO* especificadas en el campo *GMMO* de MQGMO.

El mensaje al que apunta el cursor para examinar es el que se recuperó por última vez utilizando la opción GMBRWF o GMBRWN. La llamada falla si no se ha emitido ninguna de estas llamadas

para esta cola desde que se abrió, o si el mensaje que estaba bajo el cursor para examinar se ha recuperado desde entonces de forma destructiva.

Esta llamada no cambia la posición del cursor para examinar.

A continuación, la opción GMMUC se puede utilizar con una llamada MQGET sin examinar si es necesario, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET para no examinar utilizando el mismo descriptor de contexto *HOBJ* . Tampoco se mueve mediante una llamada MQGET para examinar que devuelve un código de terminación de CCFAIL, o un código de razón de RC2080.

Si se especifica GMBRWC con GMLK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor, para que se devuelva sin desbloquearlo y volver a bloquearlo; el mensaje permanece bloqueado.
- Si no hay ningún mensaje bloqueado, el mensaje bajo el cursor para examinar (si hay uno) se bloquea y se devuelve a la aplicación; si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica GMBRWC sin GMLK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor. Este mensaje se devuelve a la aplicación y, a continuación, se desbloquea. Puesto que el mensaje está ahora desbloqueado, no hay garantía de que pueda volver a examinarse o recuperarse de forma destructiva (otra aplicación puede recuperarlo de forma destructiva obteniendo mensajes de la cola).
- Si no hay ningún mensaje bloqueado, el mensaje bajo el cursor para examinar (si hay uno) se devuelve a la aplicación; si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica GMCMPM con GMBRWC, el cursor para examinar debe identificar un mensaje con un campo *MDOFF* en MQMD que sea cero. Si esta condición no se cumple, la llamada falla con el código de razón RC2246 .

La información de grupo y segmento que el gestor de colas retiene para las llamadas MQGET que examinan mensajes en la cola, está separada de la información de grupo y segmento que retiene para las llamadas MQGET que eliminan mensajes de la cola.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWF
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se ha abierto para examinar.

GMMUC

Obtener mensaje bajo el cursor para examinar.

Esta opción hace que se recupere el mensaje al que apunta el cursor para examinar, independientemente de las opciones MO* especificadas en el campo *GMMO* en MQGMO. El mensaje se elimina de la cola.

El mensaje al que apunta el cursor para examinar es el que se recuperó por última vez utilizando la opción GMBRWF o GMBRWN.

Si se especifica GMCMPM con GMMUC, el cursor para examinar debe identificar un mensaje con un campo *MDOFF* en MQMD que sea cero. Si esta condición no se cumple, la llamada falla con el código de razón RC2246 .

Esta opción no es válida con ninguna de las opciones siguientes:

- GMBRWF
- GMBRWC
- GMBRWN
- GMUNLK

También es un error si la cola no se ha abierto tanto para examinar como para entrada. Si el cursor para examinar no apunta actualmente a un mensaje recuperable, la llamada MQGET devuelve un error.

Opciones de bloqueo: Las opciones siguientes están relacionadas con el bloqueo de mensajes en la cola:

GMLK

Bloquear mensaje.

Esta opción bloquea el mensaje que se examina, de modo que el mensaje se vuelve invisible para cualquier otro descriptor de contexto abierto para la cola. La opción sólo se puede especificar si también se especifica una de las opciones siguientes:

- GMBRWF
- GMBRWN
- GMBRWC

Sólo se puede bloquear un mensaje por descriptor de contexto de cola, pero puede ser un mensaje lógico o un mensaje físico:

- Si se especifica GMCMPM, todos los segmentos de mensajes que componen el mensaje lógico se bloquean en el descriptor de contexto de cola (si todos están presentes en la cola y disponibles para su recuperación).
- Si no se especifica GMCMPM, sólo se bloquea un único mensaje físico en el descriptor de contexto de cola. Si este mensaje es un segmento de un mensaje lógico, el segmento bloqueado impide que otras aplicaciones utilicen GMCMPM para recuperar o examinar el mensaje lógico.

El mensaje bloqueado es siempre el que está bajo el cursor para examinar, y el mensaje se puede eliminar de la cola mediante una llamada MQGET posterior que especifique la opción GMMUC. Otras llamadas MQGET que utilizan el descriptor de contexto de cola también pueden eliminar el mensaje (por ejemplo, una llamada que especifica el identificador de mensaje del mensaje bloqueado).

Si la llamada devuelve el código de terminación CCFAIL, o CCWARN con el código de razón RC2080, no se bloquea ningún mensaje.

Si la aplicación decide no eliminar el mensaje de la cola, el bloqueo lo libera:

- Emitiendo otra llamada MQGET para este descriptor de contexto, con GMBRWF o GMBRWN especificados (con o sin GMLK); el mensaje se desbloquea si la llamada se completa con CCOK o CCWARN, pero permanece bloqueado si la llamada se completa con CCFAIL. No obstante, se aplican las excepciones siguientes:
  - El mensaje no se desbloquea si se devuelve CCWARN con RC2080.
  - El mensaje se desbloquea si se devuelve CCFAIL con RC2033.

  Si también se especifica GMLK, el mensaje devuelto se bloquea. Si no se especifica GMLK, no hay ningún mensaje bloqueado después de la llamada.

  Si se especifica GMWT y no hay ningún mensaje disponible inmediatamente, el desbloqueo del mensaje original se produce antes del inicio de la espera (siempre que la llamada esté libre de errores).
- Emitiendo otra llamada MQGET para este descriptor de contexto, con GMBRWC (sin GMLK); el mensaje se desbloquea si la llamada se completa con CCOK o CCWARN, pero permanece bloqueado si la llamada se completa con CCFAIL. Sin embargo, se aplica la siguiente excepción:

– El mensaje no se desbloquea si se devuelve CCWARN con RC2080.

- Emitiendo otra llamada MQGET para este descriptor de contexto con GMUNLK.
- Emitiendo una llamada MQCLOSE para este descriptor de contexto (ya sea explícita o implícitamente por la finalización de la aplicación).

No es necesaria ninguna opción de apertura especial para especificar esta opción, que no sea OOBRW, que es necesaria para especificar la opción de examen que la acompaña.

Esta opción no es válida con ninguna de las opciones siguientes:

- GMSYP
- GMPSYP
- GMUNLK

GMUNLK
Desbloquear mensaje.

El mensaje que se va a desbloquear debe haber sido bloqueado previamente por una llamada MQGET con la opción GMLK. Si no hay ningún mensaje bloqueado para este descriptor de contexto, la llamada se completa con CCWARN y RC2209 .

Los parámetros **MSGDSC**, **BUFLEN**, **BUFFER**y **DATLEN** no se comprueban ni modifican si se especifica GMUNLK. No se devuelve ningún mensaje en *BUFFER*.

No es necesaria ninguna opción de apertura especial para especificar esta opción (aunque se necesita OOBRW para emitir la petición de bloqueo en primer lugar).

Esta opción no es válida con ninguna opción excepto la siguiente:

- GMNWT
- GMNSYP

Ambas opciones se asumen tanto si se especifican como si no.

Opciones de datos de mensaje: Las opciones siguientes están relacionadas con el proceso de los datos de mensaje cuando el mensaje se lee de la cola:

GMATM
Permitir truncamiento de datos de mensaje.

Si el almacenamiento intermedio de mensajes es demasiado pequeño para contener el mensaje completo, esta opción permite que la llamada MQGET llene el almacenamiento intermedio con la mayor cantidad de mensajes que pueda contener el almacenamiento intermedio, emita un código de finalización de aviso y complete su proceso. Esto implica lo siguiente:

- Al examinar mensajes, el cursor para examinar se avanza al mensaje devuelto.
- Al eliminar mensajes, el mensaje devuelto se elimina de la cola.
- Se devuelve el código de razón RC2079 si no se produce ningún otro error.

Sin esta opción, el almacenamiento intermedio se sigue llenando con la mayor parte del mensaje que puede contener, se emite un código de finalización de aviso, pero el proceso no se ha completado. Esto implica lo siguiente:

- Al examinar mensajes, el cursor para examinar no está avanzado.
- Al eliminar mensajes, el mensaje no se elimina de la cola.
- Se devuelve el código de razón RC2080 si no se produce ningún otro error.

GMCONV
Convertir datos de mensaje.

Esta opción solicita que los datos de aplicación del mensaje se conviertan, para ajustarse a los valores *MDCSI* y *MDENC* especificados en el parámetro **MSGDSC** de la llamada MQGET, antes de que los datos se copien en el parámetro **BUFFER** .

El campo *MDFMT* especificado cuando se colocó el mensaje es asumido por el proceso de conversión para identificar la naturaleza de los datos en el mensaje. La conversión de los datos de mensaje la realiza el gestor de colas para los formatos incorporados y una salida escrita por el usuario para otros formatos.

- Si la conversión se realiza correctamente, los campos *MDCSI* y *MDENC* especificados en el parámetro **MSGDSC** no se modifican en la devolución de la llamada MQGET.
- Si la conversión no se puede realizar correctamente (pero la llamada MQGET se completa sin errores), los datos del mensaje se devuelven sin convertir y los campos *MDCSI* y *MDENC* de *MSGDSC* se establecen en los valores del mensaje sin convertir. El código de terminación es CCWARN en este caso.

Por lo tanto, en cualquier caso, estos campos describen el identificador de juego de caracteres y la codificación de los datos de mensaje que se devuelven en el parámetro **BUFFER** .

Consulte el campo *MDFMT* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener una lista de nombres de formato para los que el gestor de colas realiza la conversión.

Opciones de grupo y segmento: Las opciones siguientes están relacionadas con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Estas definiciones pueden ser de ayuda para comprender las opciones:

Mensaje físico
Esta es la unidad de información más pequeña que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Generalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (campo*MDMID* en MQMD), aunque el gestor de colas no los impone.

Mensaje lógico
Se trata de una única unidad de información de aplicación. En ausencia de restricciones del sistema, un mensaje lógico sería el mismo que un mensaje físico. Pero cuando los mensajes lógicos son grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados segmentos.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (campo*MDGID* en MQMD) y el mismo número de secuencia de mensaje (campo*MDSEQ* en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento (campo*MDOFF* en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Puesto que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo, aunque en este caso sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos para los que la aplicación emisora ha inhibido la segmentación tienen un identificador de grupo nulo (GINONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes
Es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes para el número de secuencia de mensaje, que es un entero en el rango de 1 a n, donde n es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

GLOGO
Los mensajes en grupos y segmentos de mensajes lógicos se devuelven en orden lógico.

Esta opción controla el orden en que las llamadas MQGET sucesivas devuelven los mensajes para el descriptor de contexto de cola. La opción debe especificarse en cada una de esas llamadas para tener un efecto.

Si se especifica GMLOGO para llamadas MQGET sucesivas para el descriptor de contexto de cola, los mensajes de los grupos se devuelven en el orden indicado por sus números de secuencia de mensajes, y los segmentos de mensajes lógicos se devuelven en el orden indicado por sus desplazamientos de segmento. Este orden puede ser diferente del orden en el que se producen estos mensajes y segmentos en la cola.

Nota: La especificación de GMLOGO no tiene consecuencias adversas en los mensajes que no pertenecen a grupos y que no son segmentos. En efecto, estos mensajes se tratan como si cada uno perteneciera a un grupo de mensajes que consta de un solo mensaje. Por lo tanto, es perfectamente seguro especificar GMLOGO al recuperar mensajes de colas que pueden contener una mezcla de mensajes en grupos, segmentos de mensajes y mensajes no segmentados, no en grupos.

Para devolver los mensajes en el orden necesario, el gestor de colas conserva la información de grupo y segmento entre llamadas MQGET sucesivas. Esta información identifica el grupo de mensajes actual y el mensaje lógico actual para el descriptor de contexto de cola, la posición actual dentro del grupo y el mensaje lógico, y si los mensajes se están recuperando dentro de una unidad de trabajo. Puesto que el gestor de colas conserva esta información, la aplicación no necesita establecer la información de grupo y segmento antes de cada llamada MQGET. Específicamente, significa que la aplicación no necesita establecer los campos *MDGID*, *MDSEQ*y *MDOFF* en MQMD. Sin embargo, la aplicación no necesita establecer correctamente la opción GMSYP o GMNSYP en cada llamada.

Cuando se abre la cola, no hay ningún grupo de mensajes actual ni ningún mensaje lógico actual. Un grupo de mensajes se convierte en el grupo de mensajes actual cuando la llamada MQGET devuelve un mensaje que tiene el distintivo MFMIG. Con GMLOGO especificado en llamadas sucesivas, ese grupo permanece en el grupo actual hasta que se devuelve un mensaje que tiene:

- MFLMIG sin MFSEG (es decir, el último mensaje lógico del grupo no está segmentado), o
- MFLMIG con MFLSEG (es decir, el mensaje devuelto es el último segmento del último mensaje lógico del grupo).

Cuando se devuelve un mensaje de este tipo, el grupo de mensajes termina y, al finalizar correctamente la llamada MQGET, ya no hay un grupo actual. De forma similar, un mensaje lógico se convierte en el mensaje lógico actual cuando la llamada MQGET devuelve un mensaje que tiene el distintivo MFSEG, y ese mensaje lógico termina cuando se devuelve el mensaje que tiene el distintivo MFLSEG.

Si no se especifica ningún criterio de selección, las llamadas MQGET sucesivas devuelven (en el orden correcto) los mensajes para el primer grupo de mensajes de la cola y, a continuación, los mensajes para el segundo grupo de mensajes, y así sucesivamente, hasta que no haya más mensajes disponibles. Es posible seleccionar los grupos de mensajes concretos devueltos especificando una o más de las opciones siguientes en el campo *GMMO* :

- MOMSGI
- MOCORI
- MOGRPI

Sin embargo, estas opciones sólo son efectivas cuando no hay ningún grupo de mensajes actual o mensaje lógico; consulte el campo *GMMO* descrito en este tema.

La Tabla 702 en la página 1112 muestra los valores de los campos *MDMID*, *MDCID*, *MDGID*, *MDSEQ*y *MDOFF* que el gestor de colas busca al intentar encontrar un mensaje que devolver en la llamada MQGET. Esto se aplica tanto a la eliminación de mensajes de la cola como a la exploración de mensajes en la cola. Las columnas de la tabla tienen los significados siguientes:

LOG ORD
Indica si se ha especificado la opción GMLOGO en la llamada.

Cur grp
Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg
Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas
Mostrar los valores que busca el gestor de colas. "Anterior" indica el valor devuelto para el campo en el mensaje anterior para el descriptor de contexto de cola.

Tabla 702. Opciones MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos

Opciones especificadas	**Estado de grupo y mensaje lógico antes de la llamada**		**Valores que el gestor de colas busca**				
LOG ORD	**Cur grp**	**Cur log msg**	*MDMID*	*MDCID*	*MDGID*	*MDSEQ*	*MDOFF*
Sí	No	No	Controlado por *GMMO*	Controlado por *GMMO*	Controlado por *GMMO*	1	0
Sí	No	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior
Sí	Sí	No	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	Cualquiera de los dos	Cualquiera de los dos	Controlado por *GMMO*	Controlado por *GMMO*	Controlado por *GMMO*	Controlado por *GMMO*	Controlado por *GMMO*

Cuando varios grupos de mensajes están presentes en la cola y son elegibles para la devolución, los grupos se devuelven en el orden determinado por la posición en la cola del primer segmento del primer mensaje lógico de cada grupo (es decir, los mensajes físicos que tienen números de secuencia de mensaje de 1, y desplazamientos de 0, determinan el orden en el que se devuelven los grupos elegibles).

La opción GMLOGO afecta a las unidades de trabajo de la siguiente manera:

- Si el primer mensaje lógico o segmento de un grupo se recupera dentro de una unidad de trabajo, todos los demás mensajes lógicos y segmentos del grupo deben recuperarse dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola. Sin embargo, no es necesario recuperarlos dentro de la misma unidad de trabajo. Esto permite que un grupo de mensajes que consta de muchos mensajes físicos se divida entre dos o más unidades de trabajo consecutivas para el descriptor de contexto de cola.
- Si el primer mensaje lógico o segmento de un grupo no se recupera dentro de una unidad de trabajo, ninguno de los otros mensajes lógicos y segmentos del grupo se puede recuperar dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola.

Si no se cumplen estas condiciones, la llamada MQGET falla con el código de razón RC2245 .

Cuando se especifica GMLOGO, el MQGMO proporcionado en la llamada MQGET no debe ser menor que GMVER2y el MQMD no debe ser menor que MDVER2. Si no se cumple esta condición, la llamada falla con el código de razón RC2256 o RC2257 , según corresponda.

Si no se especifica GMLOGO para llamadas MQGET sucesivas para el descriptor de contexto de cola, los mensajes se devuelven sin tener en cuenta si pertenecen a grupos de mensajes o si son segmentos de mensajes lógicos. Esto significa que los mensajes o segmentos de un grupo o mensaje lógico determinado pueden devolverse desordenados, o pueden estar entremezclados con mensajes o segmentos de otros grupos o mensajes lógicos, o con mensajes que no están en grupos y no son segmentos. En esta situación, los mensajes concretos que devuelven las llamadas MQGET sucesivas se controlan mediante las opciones MO* especificadas en dichas llamadas (consulte el campo *GMMO* que se describe en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener detalles de estas opciones).

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico en el medio, después de que se haya producido una anomalía del sistema. Cuando se reinicia el sistema, la aplicación puede establecer los campos *MDGID*, *MDSEQ*, *MDOFF*y *GMMO* en los valores adecuados y, a continuación, emitir la llamada MQGET con GMSYP o GMNSYP establecidos según sea necesario, pero sin especificar GMLOGO. Si esta llamada es satisfactoria, el gestor de colas conserva la información de grupo y segmento, y las llamadas MQGET subsiguientes que utilizan ese descriptor de contexto de cola pueden especificar GMLOGO como normal.

La información de grupo y segmento que el gestor de colas retiene para la llamada MQGET es independiente de la información de grupo y segmento que retiene para la llamada MQPUT. Además, el gestor de colas conserva información separada para:

- Llamadas MQGET que eliminan mensajes de la cola.
- Llamadas MQGET que examinan mensajes en la cola.

Para cualquier descriptor de contexto de cola determinado, la aplicación puede combinar llamadas MQGET que especifiquen GMLOGO con llamadas MQGET que no lo hagan, pero se deben tener en cuenta los puntos siguientes:

- Si no se especifica GMLOGO, cada llamada MQGET satisfactoria hace que el gestor de colas establezca la información de grupo y segmento guardada en los valores correspondientes al mensaje devuelto; esto sustituye la información de grupo y segmento existente retenida por el gestor de colas para el descriptor de contexto de cola. Sólo se modifica la información adecuada para la acción de la llamada (examinar o eliminar).
- Si no se especifica GMLOGO, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; sin embargo, la llamada podría tener éxito con un código de terminación CCWARN. La Tabla 703 en la página 1113 muestra los distintos casos que se pueden presentar. En estos casos, si el código de terminación no es CCOK, el código de razón es uno de los siguientes:
  - RC2241
  - RC2242
  - RC2245

  Nota: El gestor de colas no comprueba la información de grupo y segmento al examinar una cola, o al cerrar una cola que se ha abierto para examinar pero no para entrar; en estos casos, el código de finalización siempre es CCOK (suponiendo que no haya otros errores).

Tabla 703. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información de grupo y segmento

La llamada actual es	La llamada anterior era MQGET con GMLOGO	La llamada anterior era MQGET sin GMLOGO
MQGET con GMLOGO	CCFAIL	CCFAIL
MQGET sin GMLOGO	CCWARN	CCOK

Tabla 703. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información de grupo y segmento (continuación)		
La llamada actual es	**La llamada anterior era MQGET con GMLOGO**	**La llamada anterior era MQGET sin GMLOGO**
MQCLOSE con un grupo o mensaje lógico sin terminar	CCWARN	CCOK

Las aplicaciones que simplemente desean recuperar mensajes y segmentos en orden lógico se recomiendan para especificar GMLOGO, ya que esta es la opción más sencilla de utilizar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, es posible que las aplicaciones especializadas necesiten más control del que proporciona la opción GMLOGO, y esto se puede lograr no especificando esa opción. Si esto se hace, la aplicación debe asegurarse de que los campos *MDMID*, *MDCID*, *MDGID*, *MDSEQ*y *MDOFF* en MQMD, y las opciones MO* en *GMMO* en MQGMO, se establecen correctamente, antes de cada llamada MQGET.

Por ejemplo, una aplicación que desea reenviar mensajes físicos que recibe, sin tener en cuenta si estos mensajes están en grupos o segmentos de mensajes lógicos, no debe especificar GMLOGO. Esto se debe a que en una red compleja con varias vías de acceso entre el envío y la recepción de gestores de colas, los mensajes físicos pueden llegar desordenados. Al no especificar GMLOGO y el PMLOGO correspondiente en la llamada MQPUT, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llegue, sin tener que esperar a que llegue el siguiente en orden lógico.

GMLOGO se puede especificar con cualquiera de las otras opciones GM*, y con varias de las opciones MO* en las circunstancias adecuadas.

GMCMPM

Sólo se pueden recuperar los mensajes lógicos completos.

Esta opción especifica que la llamada MQGET sólo puede devolver un mensaje lógico completo. Si el mensaje lógico está segmentado, el gestor de colas vuelve a ensamblar los segmentos y devuelve el mensaje lógico completo a la aplicación; el hecho de que el mensaje lógico se haya segmentado no es aparente para la aplicación que lo recupera.

Nota: Esta es la única opción que hace que el gestor de colas vuelva a ensamblar segmentos de mensajes. Si no se especifica, los segmentos se devuelven individualmente a la aplicación si están presentes en la cola (y cumplen los otros criterios de selección especificados en la llamada MQGET). Por lo tanto, las aplicaciones que no deseen recibir segmentos individuales deben especificar siempre GMCMPM.

Para utilizar esta opción, la aplicación debe proporcionar un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo o especificar la opción GMATM.

Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y aún no han llegado), especificar GMCMPM impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos de mensajes siguen contribuyendo al valor del atributo de cola **CurrentQDepth** ; esto significa que es posible que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero.

Para los mensajes persistentes, el gestor de colas puede volver a ensamblar los segmentos sólo dentro de una unidad de trabajo:

- Si la llamada MQGET está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla parcialmente durante el proceso de reensamblaje, el gestor de colas restablece en la cola los segmentos que se han eliminado durante el reensamblaje. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.

- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo sólo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas restituye la unidad de trabajo.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar el reensamblaje. Si el mensaje no necesita reensamblarse, la llamada puede seguir siendo satisfactoria. Pero si el mensaje no necesita reensamblarse, la llamada falla con el código de razón RC2255 .

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para poder realizar el reensamblaje.

Cada mensaje físico que es un segmento tiene su propio descriptor de mensaje. Para los segmentos que constituyen un único mensaje lógico, la mayoría de los campos del descriptor de mensaje son los mismos para todos los segmentos del mensaje lógico; normalmente son sólo los campos *MDMID*, *MDOFF*y *MDMFL* los que difieren entre los segmentos del mensaje lógico. Sin embargo, si un segmento se coloca en una cola de mensajes no entregados en un gestor de colas intermedio, el manejador DLQ recupera el mensaje especificando la opción GMCONV, y esto puede dar como resultado que se cambie el juego de caracteres o la codificación del segmento. Si el manejador DLQ envía correctamente el segmento en su camino, es posible que el segmento tenga un juego de caracteres o una codificación que difiera de los otros segmentos del mensaje lógico cuando el segmento finalmente llegue al gestor de colas de destino.

El gestor de colas no puede volver a ensamblar en un único mensaje lógico un mensaje lógico que consta de segmentos en los que los campos *MDCSI*, *MDENC*o ambos difieren. En su lugar, el gestor de colas vuelve a ensamblar y devuelve los primeros segmentos consecutivos al principio del mensaje lógico que tienen los mismos identificadores de juego de caracteres y codificaciones, y la llamada MQGET se completa con el código de terminación CCWARN y el código de razón RC2243 o RC2244 , según corresponda. Esto sucede independientemente de si se especifica GMCONV. Para recuperar los segmentos restantes, la aplicación debe volver a emitir la llamada MQGET sin la opción GMCMPM, recuperando los segmentos uno por uno. GMLOGO se puede utilizar para recuperar los segmentos restantes en orden.

También es posible para una aplicación que coloca segmentos para establecer otros campos en el descriptor de mensaje en valores que difieren entre segmentos. Sin embargo, esto no tiene ninguna ventaja si la aplicación receptora utiliza GMCMPM para recuperar el mensaje lógico. Cuando el gestor de colas vuelve a ensamblar un mensaje lógico, devuelve en el descriptor de mensaje los valores del descriptor de mensaje para el primer segmento; la única excepción es el campo *MDMFL* , que el gestor de colas establece para indicar que el mensaje reensamblado es el único segmento.

Si se especifica GMCMPM para un mensaje de informe, el gestor de colas realiza un proceso especial. El gestor de colas comprueba la cola para ver si todos los mensajes de informe de ese tipo de informe relacionados con los distintos segmentos del mensaje lógico están presentes en la cola. Si lo son, se pueden recuperar como un solo mensaje especificando GMCMPM. Para que esto sea posible, los mensajes de informe deben ser generados por un gestor de colas o MCA que soporte la segmentación, o la aplicación de origen debe solicitar al menos 100 bytes de datos de mensaje (es decir, se deben especificar las opciones RO* D o RO* F adecuadas). Si hay menos de la cantidad completa de datos de aplicación para un segmento, los bytes que faltan se sustituyen por nulos en el mensaje de informe devuelto.

Si se especifica GMCMPM con GMMUC o GMBRWC, el cursor para examinar debe estar situado en un mensaje con un campo *MDOFF* en MQMD que tenga un valor de 0. Si esta condición no se cumple, la llamada falla con el código de razón RC2246 .

GMCMPM implica GMASGA, por lo que no es necesario especificarlo.

GMCMPM se puede especificar con cualquiera de las otras opciones GM* aparte de GMPSYP, y con cualquiera de las opciones MO* aparte de MOOFFS.

GMAMSA

Todos los mensajes del grupo deben estar disponibles.

Esta opción especifica que los mensajes de un grupo pasan a estar disponibles para la recuperación sólo cuando todos los mensajes del grupo están disponibles. Si la cola contiene grupos de mensajes con algunos de los mensajes que faltan (quizás porque se han retrasado en la red y todavía no han llegado), especificar GMAMSA impide la recuperación de mensajes que pertenecen a grupos incompletos. Sin embargo, estos mensajes siguen contribuyendo al valor del atributo de cola **CurrentQDepth** ; esto significa que es posible que no haya grupos de mensajes recuperables, aunque **CurrentQDepth** sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón RC2033 después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de GMAMSA depende de si también se especifica GMLOGO:

- Si se especifican ambas opciones, GMAMSA sólo afecta cuando no hay ningún grupo actual o mensaje lógico. Si hay un grupo actual o un mensaje lógico, se ignora GMAMSA. Esto significa que GMAMSA puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica GMAMSA sin GMLOGO, GMAMSA siempre tiene un efecto. Esto significa que la opción debe desactivarse después de que el primer mensaje del grupo se haya eliminado de la cola, para poder eliminar los mensajes restantes del grupo.

La finalización satisfactoria de una llamada MQGET que especifica GMAMSA significa que en el momento en que se emitió la llamada MQGET, todos los mensajes del grupo estaban en la cola. Sin embargo, tenga en cuenta que otras aplicaciones todavía pueden eliminar mensajes del grupo (el grupo no está bloqueado para la aplicación que recupera el primer mensaje del grupo).

Si no se especifica esta opción, los mensajes que pertenecen a grupos se pueden recuperar incluso cuando el grupo está incompleto.

GMAMSA implica GMASGA, por lo que no es necesario especificarlo.

GMAMSA se puede especificar con cualquiera de las otras opciones GM* y con cualquiera de las opciones MO*.

GMASGA

Todos los segmentos de un mensaje lógico deben estar disponibles.

Esta opción especifica que los segmentos de un mensaje lógico pasan a estar disponibles para la recuperación sólo cuando todos los segmentos del mensaje lógico están disponibles. Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y todavía no han llegado), especificar GMASGA impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos siguen contribuyendo al valor del atributo de cola **CurrentQDepth** ; esto significa que es posible que no haya mensajes lógicos recuperables, aunque **CurrentQDepth** sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón RC2033 después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de GMASGA depende de si también se especifica GMLOGO:

- Si se especifican ambas opciones, GMASGA sólo tiene efecto cuando no hay ningún mensaje lógico actual. Si hay un mensaje lógico actual, se ignora GMASGA. Esto significa que GMASGA puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica GMASGA sin GMLOGO, GMASGA siempre tiene efecto. Esto significa que la opción debe desactivarse después de que el primer segmento del mensaje lógico se haya eliminado de la cola, para poder eliminar los segmentos restantes del mensaje lógico.

Si no se especifica esta opción, los segmentos de mensaje se pueden recuperar incluso cuando el mensaje lógico está incompleto.

Mientras que tanto GMCMPM como GMASGA requieren que todos los segmentos estén disponibles antes de que cualquiera de ellos pueda ser recuperado, el primero devuelve el

mensaje completo, mientras que el segundo permite que los segmentos sean recuperados uno por uno.

Si se especifica GMASGA para un mensaje de informe, el gestor de colas realiza un proceso especial. El gestor de colas comprueba la cola para ver si hay al menos un mensaje de informe para cada uno de los segmentos que componen el mensaje lógico completo. Si existe, se cumple la condición GMASGA. Sin embargo, el gestor de colas no comprueba el tipo de los mensajes de informe presentes, por lo que puede haber una combinación de tipos de informe en los mensajes de informe relacionados con los segmentos del mensaje lógico. Como resultado, el éxito de GMASGA no implica que GMCMPM tenga éxito. Si hay una combinación de tipos de informe presentes para los segmentos de un mensaje lógico determinado, estos mensajes de informe se deben recuperar uno por uno.

GMASGA se puede especificar con cualquiera de las otras opciones GM* y con cualquiera de las opciones MO*.

Opción predeterminada: si ninguna de las opciones descritas es necesaria, se puede utilizar la opción siguiente:

GMNONE
No se ha especificado ninguna opción.

Este valor puede utilizarse para indicar que no se ha especificado ninguna otra opción; todas las opciones asumen sus valores predeterminados. GMNONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial del campo *GMOPT* es GMNWT.

GMRE1 (serie de caracteres de 1 byte)

Reservado.

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *GMVER* es menor que GMVER2.

GMRL (entero con signo de 10 dígitos)

Longitud de datos de mensaje devueltos (bytes).

Es un campo de salida establecido por el gestor de colas en la longitud en bytes de los datos de mensaje devueltos por la llamada MQGET en el parámetro **BUFFER** . Si el gestor de colas no da soporte a esta prestación, *GMRL* se establece en el valor RLUNDF.

Cuando los mensajes se convierten entre codificaciones o juegos de caracteres, los datos del mensaje a veces pueden cambiar de tamaño. A la devolución de la llamada MQGET:

- Si *GMRL* no es RLUNDF, *GMRL* proporciona el número de bytes de datos de mensaje devueltos.
- Si *GMRL* tiene el valor RLUNDF, el número de bytes de datos de mensaje devueltos normalmente lo proporciona el menor de *BUFLEN* y *DATLEN*, pero puede ser menor que este valor si la llamada MQGET se completa con el código de razón RC2079 . Si esto sucede, los bytes insignificantes del parámetro **BUFFER** se establecen en nulos.

Se define el siguiente valor especial:

RLUNDF
No se ha definido la longitud de los datos devueltos.

El valor inicial de este campo es RLUNDF. Este campo se ignora si *GMVER* es menor que GMVER3.

GMRQN (serie de caracteres de 48 bytes)

Nombre resuelto de la cola de destino.

Es un campo de salida que el gestor de colas establece en el nombre local de la cola de la que se ha recuperado el mensaje, tal como se ha definido en el gestor de colas local. Esto es diferente del nombre utilizado para abrir la cola si:

- Se ha abierto una cola alias (en cuyo caso, se devuelve el nombre de la cola local a la que se ha resuelto el alias), o
- Se ha abierto una cola modelo (en cuyo caso, se devuelve el nombre de la cola local dinámica).

La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

GMRS2 (serie de caracteres de 1 byte)

Reservado.

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *GMVER* es menor que GMVER4.

GMSEG (serie de caracteres de 1 byte)

Distintivo que indica si se permite una segmentación adicional para el mensaje recuperado.

Tiene uno de los siguientes valores:

SEGIHB
Segmentación no permitida.

SEGALO
Segmentación permitida.

Se trata de un campo de salida. El valor inicial de este campo es SEGIHB. Este campo se ignora si *GMVER* es menor que GMVER2.

GMSG1 (entero con signo de 10 dígitos)

Señal.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

GMSG2 (entero con signo de 10 dígitos)

Identificador de señal.

Se trata de un campo reservado; su valor no es significativo.

GMSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

GMSIDV
Identificador de la estructura de opciones de obtención de mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es GMSIDV.

GMSST (serie de caracteres de 1 byte)

Distintivo que indica si el mensaje recuperado es un segmento de un mensaje lógico.

Tiene uno de los siguientes valores:

SSNSEG
El mensaje no es un segmento.

SSEG
El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

SSLSEG
El mensaje es el último segmento del mensaje lógico.

También es el valor devuelto si el mensaje lógico consta de un solo segmento.

Este campo es un campo de salida. El valor inicial de este campo es SSNSEG. Este campo se ignora si *GMVER* es menor que GMVER2.

GMTOK (serie de bits de 16 bytes)

Señal de mensaje.

Se trata de un campo reservado; su valor no es significativo. Se define el siguiente valor especial:

MTKNON
Ninguna señal de mensaje.

El valor es cero binario para la longitud del campo.

La longitud de este campo la proporciona LNMTOK. El valor inicial de este campo es MTKNON. Este campo se ignora si *GMVER* es menor que GMVER3.

GMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

GMVER1
Estructura de opciones de obtención de mensaje Version-1 .

GMVER2
Estructura de opciones de obtención de mensaje Version-2 .

GMVER3
Estructura de opciones de obtención de Version-3 .

GMVER4
Estructura de opciones de obtención de mensaje Version-4 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

GMVERC
Versión actual de la estructura de opciones de obtención de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es GMVER1.

GMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

GMVER1
Estructura de opciones de obtención de mensaje Version-1 .

GMVER2
Estructura de opciones de obtención de mensaje Version-2 .

GMVER3
Estructura de opciones de obtención de Version-3 .

GMVER4
Estructura de opciones de obtención de mensaje Version-4 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

GMVERC
Versión actual de la estructura de opciones de obtención de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es GMVER1.

GMWI (entero con signo de 10 dígitos)

Intervalo de espera.

Es el tiempo aproximado, expresado en milisegundos, que la llamada MQGET espera a que llegue un mensaje adecuado (es decir, un mensaje que cumpla los criterios de selección especificados en el parámetro **MSGDSC** de la llamada MQGET; consulte el campo *MDMID* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener más detalles). Si no ha llegado ningún mensaje adecuado después de que haya transcurrido este tiempo, la llamada se completa con CCFAIL y el código de razón RC2033.

GMWI se utiliza con la opción GMWT. Se ignora si no se especifica esta opción. Si se especifica, *GMWI* debe ser mayor o igual que cero, o el siguiente valor especial:

WIULIM
Intervalo de espera ilimitado.

El valor inicial de este campo es 0.

Valores iniciales

Tabla 704. Valores iniciales de los campos en MQGMO

Nombre de campo	Nombre de constante	Valor de constante
GMSID	GMSIDV	'GMO¬'
GMVER	GMVER1	1
GMOPT	GMNWT	0
GMWI	Ninguna	0
GMSG1	Ninguna	0
GMSG2	Ninguna	0
GMRQN	Ninguna	Espacios en blanco
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG	'¬'
GMSST	SSNSEG	'¬'
GMSEG	SEGIHB	'¬'
GMRE1	Ninguna	'¬'
GMTOK	MTKNON	Nulos
GMRL	RLUNDF	-1
GMRS2	Ninguna	'¬'
GMMH	HMNONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (cabecera de información deIMS) en IBM i

La estructura MQIIH describe la información que debe estar presente al principio de un mensaje enviado al puente IMS a través de IBM MQ for z/OS.

Visión general

Nombre de formato: FMIMS.

Conjunto de caracteres y codificación: las condiciones especiales se aplican al juego de caracteres y la codificación utilizados para la estructura MQIIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente IMS deben proporcionar una estructura MQIIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQIIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQIIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; la conversión de MQIIH la realiza el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola puente IMS .

  Nota: Hay una excepción a esto. Si el gestor de colas propietario de la cola de puente IMS utiliza CICS para la gestión de colas distribuidas, MQIIH debe estar en el juego de caracteres y la codificación del gestor de colas propietario de la cola de puente IMS .
- Los datos del mensaje de aplicación que siguen a la estructura MQIIH deben estar en el mismo juego de caracteres y codificación que la estructura MQIIH. Los campos *IICSI* y *IIENC* de la estructura MQIIH no se pueden utilizar para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

  El usuario debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

- “Autenticación de passtickets para aplicaciones puente IMS” en la página 1122
- “Campos” en la página 1122
- “Valores iniciales” en la página 1125
- “Declaración RPG” en la página 1125

Autenticación de passtickets para aplicaciones puente IMS

Ahora es posible que los administradores de IBM MQ especifiquen el nombre de aplicación que se va a utilizar para autenticar passtickets, para aplicaciones puente IMS . Para ello, el nombre de aplicación se especifica como un nuevo atributo PTKTAPPL para la definición de objeto STGCLASS, como una serie alfanumérica de 1 a 8 caracteres.

Un valor en blanco significa que la autenticación se produce como con los releases anteriores de IBM MQ, es decir, no fluye ningún nombre de aplicación en la solicitud de autenticación y, en su lugar, se utiliza el valor MVSxxxx.

Un valor de 1 a 8 caracteres alfanuméricos debe seguir las reglas para los nombres de aplicación passticket tal como se describe en las publicaciones de RACF .

Los administradores de IBM MQ y los administradores de RACF deben ponerse de acuerdo sobre los nombres de aplicación válidos que se van a utilizar. El administrador de RACF debe crear un perfil en la clase PTKTDATA que proporcione acceso READ a los ID de usuario de todas las aplicaciones a las que se va a otorgar acceso. El administrador de IBM MQ debe crear o modificar las definiciones de STGCLASS necesarias que especifican el nombre de aplicación que se debe utilizar para la autenticación de passticket.

Para obtener información relacionada, consulte la publicación *Script (MQSC) Command Reference*.

Campos

La estructura MQIIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

IIAUT (serie de caracteres de 8 bytes)

Contraseña o passticket de RACF .

Es opcional; si se especifica, se utiliza con el ID de usuario en el contexto de seguridad MQMD para crear un UTOKEN que se envía a IMS para proporcionar un contexto de seguridad. Si no se especifica, el ID de usuario se utiliza sin verificación. Esto depende del valor de los conmutadores RACF , que pueden requerir la presencia de un autenticador.

Esto se ignora si el primer byte está en blanco o es nulo. Se puede utilizar el siguiente valor especial:

IUNÓN
Sin autenticación.

La longitud de este campo la proporciona LNAUTH. El valor inicial de este campo es IAUNON.

IICMT (serie de caracteres de 1 byte)

Modalidad de confirmación.

Consulte la publicación *OTMA Reference* para obtener más información sobre las modalidades de confirmación de IMS . El valor debe ser uno de los siguientes:

ICMCTS
Confirme y, a continuación, envíe.

Esta modalidad implica una doble cola de salida, pero tiempos de ocupación de región más cortos. Las transacciones de vía de acceso rápida y conversacional no se pueden ejecutar con esta modalidad.

ICMSTC
Enviar y, a continuación, confirmar.

El valor inicial de este campo es ICMCTS.

IICSI (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

IIENC (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

IIFLG (entero con signo de 10 dígitos)

Distintivos.

El valor debe ser:

IINONA
Sin distintivos.

El valor inicial de este campo es IINONE.

IIFMT (serie de caracteres de 8 bytes)

Nombre de formato IBM MQ de los datos que siguen a MQIIH.

Especifica el nombre de formato IBM MQ de los datos que siguen a la estructura MQIIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

IILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQIIH.

El valor debe ser:

IILEN1
Longitud de la estructura de cabecera de información de IMS .

El valor inicial de este campo es IILEN1.

IILTO (serie de caracteres de 8 bytes)

Alteración temporal de terminal lógico.

Se coloca en el campo PCB de E/S. Es opcional; si no se especifica, se utiliza el nombre de TPIPE. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona LNLTOV. El valor inicial de este campo es de 8 caracteres en blanco.

IIMMN (serie de caracteres de 8 bytes)

Nombre de correlación de servicios de formato de mensaje.

Se coloca en el campo PCB de E/S. Es opcional. En la entrada representa el MID, en la salida representa el MOD. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona LNMFMN. El valor inicial de este campo es de 8 caracteres en blanco.

IIRFM (serie de caracteres de 8 bytes)

Nombre de formato IBM MQ del mensaje de respuesta.

Es el nombre de formato IBM MQ del mensaje de respuesta que se enviará en respuesta al mensaje actual. Las reglas para codificarlo son las mismas que para el campo *MDFMT* en MQMD.

La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

IIRSV (serie de caracteres de 1 byte)

Reservado.

Es un campo reservado; debe estar en blanco.

IISEC (serie de caracteres de 1 byte)

Ámbito de seguridad.

Esto indica el proceso de seguridad de IMS necesario. Los valores siguientes están definidos:

ISSCHK
Compruebe el ámbito de seguridad.

Un ACEE se crea en la región de control, pero no en la región dependiente.

ISSFUL
Ámbito de seguridad completo.

Un ACEE almacenado en memoria caché se crea en la región de control y un ACEE no almacenado en memoria caché se crea en la región dependiente. Si utiliza ISSFUL, debe asegurarse de que el ID de usuario para el que se crea el ACEE tenga acceso a los recursos utilizados en la región dependiente.

Si no se especifican ISSCHK e ISSFUL para este campo, se presupone ISSCHK.

El valor inicial de este campo es ISSCHK.

IISID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

IISID
Identificador de la estructura de cabecera de información de IMS .

El valor inicial de este campo es IISIDV.

IITID (serie de bits de 16 bytes)

Identificador de instancia de transacción.

Este campo lo utilizan los mensajes de salida de IMS , por lo que se ignora en la primera entrada. Si *IITST* se establece en ITSIC, debe proporcionarse en la siguiente entrada, y en todas las entradas posteriores, para permitir que IMS correlacione los mensajes con la conversación correcta. Se puede utilizar el siguiente valor especial:

ITINÓN
No hay ningún ID de instancia de transacción.

La longitud de este campo la proporciona LNTIID. El valor inicial de este campo es ITINON.

IITST (serie de caracteres de 1 byte)

Estado de transacción.

Indica el estado de conversación de IMS . Esto se ignora en la primera entrada porque no existe ninguna conversación. En entradas posteriores, indica si una conversación está activa o no. En la salida se establece mediante IMS. El valor debe ser uno de los siguientes:

ISIC
En conversación.

ITSNIC
No en conversación.

ITSARC
Devolver datos de estado de transacción en formato de arquitectura.

Este valor sólo se utiliza con el mandato IMS `/DISPLAY TRAN` . Hace que los datos de estado de transacción se devuelvan en el formato de arquitectura IMS en lugar de en el formato de caracteres. Consulte Escritura de programas de transacción IMS a través de IBM MQ para obtener más detalles.

El valor inicial de este campo es ITSNIC.

IIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

IIVER1
Número de versión para la estructura de cabecera de información de IMS .

La constante siguiente especifica el número de versión de la versión actual:

IIVERC
Versión actual de la estructura de cabecera de información de IMS .

El valor inicial de este campo es IIVER1.

Valores iniciales

Tabla 705. Valores iniciales de campos en MQIIH

Nombre de campo	Nombre de constante	Valor de constante
IISID	IISID	'IIH¬'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	Ninguna	0
IICSI	Ninguna	0
IIFMT	FMNONE	Espacios en blanco
IIFLG	IINONA	0
IILTO	Ninguna	Espacios en blanco
IIMMN	Ninguna	Espacios en blanco
IIRFM	FMNONE	Espacios en blanco
IIAUT	IUNÓN	Espacios en blanco
IITID	ITINÓN	Nulos
IITST	ITSNIC	'¬'
IICMT	ICMCTS	'0'
IISEC	ISSCHK	'C'
IIRSV	Ninguna	'¬'

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Consultar opciones de propiedad de mensaje) en IBM i

La estructura MQIMPO permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes.

Visión general

Finalidad: la estructura es un parámetro de entrada en la llamada MQINQMP.

Conjunto de caracteres y codificación: Los datos de MQIMPO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- "Campos" en la página 1126
- "Valores iniciales" en la página 1132
- "Declaración RPG" en la página 1132

Campos

La estructura MQIMPO contiene los campos siguientes; los campos se describen en **orden alfabético**:

IPOPT (entero con signo de 10 dígitos)

Las opciones siguientes controlan la acción de MQINQMP. Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit). Se anotan las combinaciones de opciones que no son válidas; todas las demás combinaciones son válidas.

Opciones de datos de valor: Las opciones siguientes están relacionadas con el proceso de los datos de valor cuando se recupera la propiedad del mensaje.

ICVAL

Esta opción solicita que el valor de la propiedad se convierta para que se ajuste a los valores *IPREQCSI* y *IPREQENC* especificados antes de que la llamada MQINQMP devuelva el valor de la propiedad en el área *Value* .

- Si la conversión se realiza correctamente, los campos *IPRETCSI* y *IPRETENC* se establecen en los mismos que *IPREQCSI* y *IPREQENC* al volver de la llamada MQINQMP.

- Si la conversión falla, pero la llamada MQINQMP de lo contrario se completa sin error, el valor de propiedad se devuelve sin convertir.

  Si la propiedad es una serie, los campos *IPRETCSI* y *IPRETENC* se establecen en el juego de caracteres y la codificación de la serie no convertida.

  El código de terminación es CCWARN en este caso, con el código de razón RC2466. El cursor de propiedad se ha avanzado a la propiedad devuelta.

Si el valor de propiedad se expande durante la conversión y supera el tamaño del parámetro **Value** , el valor se devuelve sin convertir, con el código de terminación CCFAIL; el código de razón se establece en RC2469.

El parámetro **DataLength** de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Esta opción también solicita que:

- Si el nombre de propiedad contiene un comodín, y
- El campo *IPRETNAMECHRP* se inicializa con una dirección o desplazamiento para el nombre devuelto,

entonces el nombre devuelto se convierte para ajustarse a los valores *IPREQCSI* y *IPREQENC* .

- Si la conversión es satisfactoria, el campo *VSCCSID* de *IPRETNAMECHRP* y la codificación del nombre devuelto se establecen en el valor de entrada de *IPREQCSI* y *IPREQENC*.
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin error ni aviso, el nombre devuelto no se convierte. El código de terminación es CCWARN en este caso, con el código de razón RC2492.

  El cursor de propiedad se ha avanzado a la propiedad devuelta. Se devuelve RC2466 si el valor y el nombre no se convierten.

Si el nombre devuelto se expande durante la conversión y supera el tamaño del campo *VSBufsize* de *RequestedName*, la serie devuelta se deja sin convertir, con el código de terminación CCFAIL y el código de razón se establece en RC2465.

El campo *VSLength* de la estructura MQCHARV devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

IPCTYP

Esta opción solicita que el valor de la propiedad se convierta de su tipo de datos actual en el tipo de datos especificado en el parámetro **Type** de la llamada MQINQMP.

- Si la conversión es satisfactoria, el parámetro **Type** no se modifica al devolver la llamada MQINQMP.
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin error, la llamada falla con la razón RC2470. El cursor de propiedad no se ha modificado.

Si la conversión del tipo de datos hace que el valor se expanda durante la conversión y el valor convertido supera el tamaño del parámetro **Value** , el valor se devuelve sin convertir, con el código de terminación CCFAIL y el código de razón se establece en RC2469.

El parámetro **DataLength** de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Si el valor del parámetro **Type** de la llamada MQINQMP no es válido, la llamada falla con la razón RC2473.

Si la conversión de tipo de datos solicitada no está soportada, la llamada falla con la razón RC2470. Se da soporte a las siguientes conversiones de tipos de datos:

Tabla 706. Conversiones de tipos de datos soportadas

Tipo de datos de propiedad	Tipos de datos de destino soportados
TYPBOL	TYPSTR, TYPI8, TYPI16, TYPI32, TYPI64
TYPBST	TIPSTR
TYPI8	TYPSTR, TYPI16, TYPI32, TYPI64
TYPI16	TYPSTR, TYPI32, TYPI64
TYPI32	TYPSTR, TYPI64
TYPI64	TIPSTR
TYPF32	TYPSTR, TYPF64
TYPF64	TIPSTR
TIPSTR	TYPBOL, TYPI8, TYPI16, TYPI32, TYPI64, TYPF32, TYPF64
TYPNUL	Ninguna

Las normas generales que rigen las conversiones soportadas son las siguientes:

- Los valores de propiedad numéricos se pueden convertir de un tipo de datos a otro, siempre que no se pierdan datos durante la conversión.

  Por ejemplo, el valor de una propiedad con el tipo de datos TYPI32 se puede convertir en un valor con el tipo de datos TYPI64, pero no se puede convertir en un valor con el tipo de datos TYPI16.
- Un valor de propiedad de cualquier tipo de datos se puede convertir a una serie.
- Un valor de propiedad de serie se puede convertir a cualquier otro tipo de datos, siempre que la serie tenga el formato correcto para la conversión. Si una aplicación intenta convertir un valor de propiedad de serie que no tiene el formato correcto, IBM MQ devuelve el código de razón RC2472.
- Si una aplicación intenta una conversión que no está soportada, IBM MQ devuelve el código de razón RC2470.

Las reglas específicas para convertir un valor de propiedad de un tipo de datos a otro son las siguientes:

- Al convertir un valor de propiedad TYPBOL en una serie, el valor TRUE se convierte en la serie "TRUE" y el valor false se convierte en la serie "FALSE".
- Al convertir un valor de propiedad TYPBOL a un tipo de datos numérico, el valor TRUE se convierte a uno y el valor FALSE se convierte a cero.
- Al convertir un valor de propiedad de serie a un valor TYPBOL, la serie "TRUE", o "1", se convierte a TRUE, y la serie "FALSE", o "0", se convierte a FALSE.

  Tenga en cuenta que los términos "TRUE" y "FALSE" no distinguen entre mayúsculas y minúsculas.

  Cualquier otra serie no se puede convertir; IBM MQ devuelve el código de razón RC2472.
- Al convertir un valor de propiedad de serie a un valor con el tipo de datos TYPI8, TYPI16, TYPI32 o TYPI64, la serie debe tener el formato siguiente:

  ```
  [blanks][sign]digits
  ```

  El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

Después de la secuencia de caracteres de dígitos, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por sentado que la serie representa un entero decimal.

IBM MQ devuelve el código de razón RC2472 si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad de serie a un valor con el tipo de datos TYPF32 o TYPF64, la serie debe tener el formato siguiente:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

e_char
Un carácter exponente, que es "E" o "e".

e_sign
Un carácter de signo más (+) o signo menos (-) opcional para el exponente.

e_digits
Una secuencia contigua de caracteres de dígitos (0-9) para el exponente. Al menos, debe estar presente un carácter de dígito, si la serie contiene un carácter de exponente.

Detrás de la secuencia de caracteres de dígitos, o los caracteres opcionales que representan un exponente, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por supuesto que la serie representa un número de coma flotante decimal con un exponente que es una potencia de 10.

IBM MQ devuelve el código de razón RC2472 si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad numérico en una serie, el valor se convierte a la representación de serie del valor como un número decimal, no la serie que contiene el carácter ASCII para ese valor. Por ejemplo, el entero 65 se convierte a la serie "65", no a la serie "A".
- Al convertir un valor de propiedad de serie de bytes en una serie, cada byte se convierte en los dos caracteres hexadecimales que representan el byte. Por ejemplo, la matriz de bytes {0xF1, 0x12, 0x00, 0xFF} se convierte a la serie "F11200FF".

IPQLEN
Consulte el tipo y la longitud del valor de propiedad. La longitud se devuelve en el parámetro **DataLength** de la llamada MQINQMP. El valor de propiedad no se devuelve.

Si se especifica un almacenamiento intermedio *ReturnedName* , el campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El nombre de propiedad no se devuelve.

Opciones de iteración: Las opciones siguientes están relacionadas con la iteración sobre propiedades, utilizando un nombre con un carácter comodín

IPIINQF

Consultar la primera propiedad que coincide con el nombre especificado. Después de esta llamada, se establece un cursor en la propiedad que se devuelve.

Éste es el valor predeterminado.

La opción IPINQC se puede utilizar posteriormente con una llamada MQINQMP, si es necesario, para volver a consultar la misma propiedad.

Tenga en cuenta que sólo hay un cursor de propiedad; por lo tanto, si el nombre de propiedad, especificado en la llamada MQINQMP, cambia el cursor se restablece.

Esta opción no es válida con ninguna de las opciones siguientes:

IPINQN
IPINQC

IPINQN

Consulta la siguiente propiedad que coincide con el nombre especificado, continuando la búsqueda desde el cursor de propiedad. El cursor está avanzado a la propiedad que se devuelve.

Si esta es la primera llamada MQINQMP para el nombre especificado, se devuelve la primera propiedad que coincide con el nombre especificado.

La opción IPINQC se puede utilizar posteriormente con una llamada MQINQMP si es necesario, para volver a consultar la misma propiedad.

Si la propiedad bajo el cursor se ha suprimido, MQINQMP devuelve la siguiente propiedad coincidente después de la que se ha suprimido.

Si se añade una propiedad que coincide con el comodín, mientras una iteración está en curso, es posible que la propiedad se devuelva o no durante la finalización de la iteración. La propiedad se devuelve una vez que la iteración se reinicia utilizando IPINQF.

Una propiedad que coincide con el comodín que se ha suprimido, mientras la iteración estaba en curso, no se devuelve después de su supresión.

Esta opción no es válida con ninguna de las opciones siguientes:

IPIINQF
IPINQC

IPINQC

Recupere el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez, utilizando la opción IPINQF o la opción IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, cuando se especifica el descriptor de mensaje en el campo *MsgHandle* del MQGMO en una llamada MQGET, o cuando se especifica el descriptor de mensaje en los campos *OriginalMsgHandle* o *NewMsgHandle* de la estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando el cursor de propiedad todavía no se ha establecido, o si la propiedad a la que apunta el cursor de propiedad se ha suprimido, la llamada falla con el código de terminación CCFAIL y la razón RC2471.

Esta opción no es válida con ninguna de las opciones siguientes:

IPIINQF
IPINQN

Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

IPNONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

IPNONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Siempre es un campo de entrada. El valor inicial de este campo es IPINQF.

IPREQCSI (entero con signo de 10 dígitos)

El juego de caracteres en el que se va a convertir el valor de propiedad consultado si el valor es una serie de caracteres. También es el juego de caracteres en el que se convertirá *ReturnedName* cuando se especifique IPCVAL o IPCTYP.

El valor inicial de este campo es CSAPL.

IPREQENC (entero con signo de 10 dígitos)

Esta es la codificación en la que se convertirá el valor de propiedad consultado cuando se especifique IPCVAL o IPCTYP.

El valor inicial de este campo es ENNAT.

IPRE1 (entero con signo de 10 dígitos)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco.

IPRETCSI (entero con signo de 10 dígitos)

En la salida, es el juego de caracteres del valor devuelto si el parámetro **Type** de la llamada MQINQMP es TYPSTR.

Si se especifica la opción IPCVAL y la conversión ha sido satisfactoria, el campo *ReturnedCCSID* , en el retorno, es el mismo valor que el valor pasado.

El valor inicial de este campo es cero.

IPRETENC (entero con signo de 10 dígitos)

En la salida, es la codificación del valor devuelto.

Si se especifica la opción IPCVAL y la conversión ha sido satisfactoria, el campo *ReturnedEncoding* , en el retorno, es el mismo valor que el valor pasado.

El valor inicial de este campo es ENNAT.

IPRETNAMCHRP (entero con signo de 10 dígitos)

El nombre real de la propiedad consultado.

En la entrada, se puede pasar un almacenamiento intermedio de serie utilizando el campo *VSPtr* o *VSOffset* de la estructura MQCHARV. La longitud del almacenamiento intermedio de serie se especifica utilizando el campo *VSBufsize* de la estructura MQCHARV.

Al volver de la llamada MQINQMP, el almacenamiento intermedio de serie se completa con el nombre de la propiedad que se ha consultado, siempre que el almacenamiento intermedio de serie sea lo suficientemente largo como para contener completamente el nombre. El campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El campo *VSCCSID* de la estructura MQCHARV se rellena para indicar el juego de caracteres del nombre devuelto, si la conversión del nombre ha fallado o no.

Es un campo de entrada/salida. El valor inicial de este campo es MQCHARV_DEFAULT.

IPSID (entero con signo de 10 dígitos)

Es el identificador de estructura. El valor debe ser:

IPSIDV
Identificador de la estructura de opciones de propiedad de mensaje de consulta.

Siempre es un campo de entrada. El valor inicial de este campo es IPSIDV.

IPTYP (entero con signo de 10 dígitos)

Una representación de serie del tipo de datos de la propiedad.

Si la propiedad se ha especificado en una cabecera MQRFH2 y no se reconoce el atributo MQRFH2 `dt` , este campo se puede utilizar para determinar el tipo de datos de la propiedad. *`TypeString`* se devuelve en el juego de caracteres codificado 1208 (UTF-8), y son los primeros ocho bytes del valor del atributo `dt` de la propiedad que no se ha podido reconocer

Siempre es un campo de salida. El valor inicial de este campo es la serie nula en el lenguaje de programación C y 8 caracteres en blanco en otros lenguajes de programación.

IPVER (entero con signo de 10 dígitos)

Es el número de versión de la estructura. El valor debe ser:

IPVER1
Número de versión para la estructura de opciones de propiedad de mensaje de consulta.

La constante siguiente especifica el número de versión de la versión actual:

IVERC
Versión actual de la estructura de opciones de propiedad de mensaje de consulta.

Siempre es un campo de entrada. El valor inicial de este campo es IPVER1.

Valores iniciales

Tabla 707. Valores iniciales de campos en MQIPMO

Nombre de campo	Nombre de constante	Valor de constante
IPSID	IPSIDV	`'IMPO'`
IPVER	IPVER1	1
IPOPT	IPIINQF	
IPREQENC	ENNAT	
IPREQCSI	CSAPL	
IPRETENC	ENNAT	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		vacíos

Declaración 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 (Descriptor de mensaje) en IBM i

Visión general

Finalidad: la estructura MQMD contiene la información de control que acompaña a los datos de aplicación cuando un mensaje viaja entre las aplicaciones de envío y recepción. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1 .

Versión: La versión actual de MQMD es MDVER2. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQMD soportada por el entorno, pero con el valor inicial del campo MDVER establecido en MDVER1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo MDVER en el número de versión de la versión necesaria.

Hay disponible una declaración para la estructura version-1 con el nombre MQMD1.

Conjunto de caracteres y codificación: los datos de MQMD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Si los gestores de colas emisores y receptores utilizan conjuntos de caracteres o codificaciones diferentes, los datos de MQMD se convierten automáticamente. No es necesario que la aplicación convierta MQMD.

- “Utilización de distintas versiones de MQMD” en la página 1134

- “Contexto de mensaje” en la página 1134
- “Caducidad de mensaje” en la página 1135
- “Campos” en la página 1135
- “Valores iniciales” en la página 1177
- “Declaración RPG” en la página 1178

Utilización de distintas versiones de MQMD

Un MQMD de version-2 suele ser equivalente a utilizar un MQMD de version-1 y a prefijar los datos del mensaje con una estructura MQMDE. Sin embargo, si todos los campos de la estructura MQMDE tienen sus valores predeterminados, se puede omitir MQMDE. Un MQMD version-1 más MQMDE se utilizan tal como se describe más adelante en esta sección.

- En las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD version-1 , la aplicación puede opcionalmente añadir un prefijo a los datos de mensaje con un MQMDE, estableciendo el campo MDFMT de MQMD en FMMDE para indicar que existe un MQMDE. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE.

  Nota: Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en las llamadas MQPUT y MQPUT1 . Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .
- En la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. El campo MDFMT en MQMD tendrá el valor FMMDE para indicar que hay un MQMDE presente.

Los valores predeterminados que el gestor de colas ha utilizado para los campos en MQMDE son los mismos que los valores iniciales de dichos campos, que se muestran en la Tabla 709 en la página 1177.

Cuando un mensaje está en una cola de transmisión, algunos de los campos de MQMD se establecen en valores concretos; consulte “MQXQH (cabecera de cola de transmisión) en IBM i” en la página 1274 para obtener más detalles.

Contexto de mensaje

Determinados campos de MQMD contienen el contexto del mensaje. Por norma, se utiliza para:

- El *contexto de identidad* está relacionado con la aplicación que colocó originalmente el mensaje
- El *contexto de origen* está relacionado con la aplicación que ha colocado el mensaje más recientemente
- El *contexto de usuario* está relacionado con la aplicación que colocó originalmente el mensaje.

Estas dos aplicaciones pueden ser la misma aplicación, pero también pueden ser aplicaciones diferentes (por ejemplo, cuando se reenvía un mensaje de una aplicación a otra).

Aunque el contexto de identidad y origen normalmente tiene los significados descritos anteriormente, el contenido de ambos tipos de campos de contexto en MQMD depende realmente de las opciones PM* que se especifican cuando se coloca el mensaje. Como resultado, el contexto de identidad no está necesariamente relacionado con la aplicación que colocó originalmente el mensaje, y el contexto de origen no está necesariamente relacionado con la aplicación que colocó el mensaje más recientemente- depende del diseño de la suite de aplicaciones.

Hay una clase de aplicación que nunca altera el contexto de mensaje, es decir, el agente de canal de mensajes (MCA). Los MCA que reciben mensajes de gestores de colas remotos utilizan la opción de contexto PMSETA en la llamada MQPUT o MQPUT1 . Esto permite al MCA receptor conservar exactamente el contexto de mensaje que ha viajado con el mensaje del MCA emisor. Sin embargo, el resultado es que el contexto de origen no está relacionado con la aplicación que ha colocado el mensaje más recientemente (el MCA receptor), sino que está relacionado con una aplicación anterior que ha colocado el mensaje (posiblemente la propia aplicación de origen).

Para obtener más información, consulte Contexto de mensaje.

Caducidad de mensaje

Los mensajes que han caducado en una cola cargada (una cola que se ha abierto) se eliminan automáticamente de la cola dentro de un periodo de tiempo razonable después de su caducidad. Algunas otras características nuevas de este release de IBM MQ pueden llevar a que las colas cargadas se exploren con menos frecuencia que en la versión anterior del producto, sin embargo, los mensajes caducados en las colas cargadas siempre se eliminan dentro de un periodo razonable desde su caducidad.

Campos

La estructura MQMD contiene los campos siguientes; los campos se describen en orden alfabético:

MDACC (serie de bits de 32 bytes)

Señal de contabilidad.

Forma parte del *contexto de identidad* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

MDACC permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cargue correctamente. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido.

Cuando el gestor de colas genera esta información, se establece de la siguiente manera:

- El primer byte del campo se establece en la longitud de la información de contabilidad presente en los bytes siguientes; esta longitud está en el rango de cero a 30, y se almacena en el primer byte como un entero binario.
- El segundo y los bytes subsiguientes (según lo especificado por el campo de longitud) se establecen en la información de contabilidad adecuada para el entorno.
  - z/OS En z/OS , la información de contabilidad se establece en:
    - Para el proceso por lotes z/OS , la información de contabilidad de la tarjeta JES JOB o de una sentencia ACCT de JES en la tarjeta EXEC (los separadores de coma se cambian a X'FF '). Esta información se trunca, si es necesario, a 31 bytes.
    - Para TSO, el número de cuenta del usuario.
    - Para CICS, el identificador de unidad de trabajo de LU 6.2 (UEPUOWDS) (26 bytes).
    - Para IMS, el nombre PSB de 8 caracteres concatenado con la señal de recuperación IMS de 16 caracteres.
  - IBM i En IBM i, la información de contabilidad se establece en el código de contabilidad del trabajo.
  - UNIX En UNIX, la información de contabilidad se establece en el identificador de usuario numérico, en caracteres ASCII.
  - Windows En Windows, la información de contabilidad se establece en un identificador de seguridad (SID) de Windows NT en un formato comprimido. El SID identifica de forma exclusiva el identificador de usuario almacenado en el campo *MDUID* . Cuando el SID se almacena en el campo *MDACC* , se omite la autoridad de identificador de 6 bytes (ubicada en el tercer y subsiguiente bytes del SID). Por ejemplo, si el SID de Windows NT tiene una longitud de 28 bytes, se almacenan 22 bytes de información de SID en el campo *MDACC* .
- El último byte se establece en el tipo de señal de contabilidad, uno de los valores siguientes:

  ATTCIC
  CICS Identificador LUOW.

ATTDOS
Señal de contabilidad predeterminada de PC DOS.

ATTWNT
Identificador de seguridad de Windows .

ATT400
Señal de contabilidad de IBM i .

ATTUNX
Identificador numérico de UNIX .

ATTUSR
Señal de contabilidad definida por el usuario.

ATESTADO
Tipo de señal de contabilidad desconocido.

El tipo de señal de contabilidad se establece en un valor explícito sólo en los entornos siguientes:

– AIX

– IBM i

– Solaris

– Windows

y para IBM MQ MQI clients conectados a estos sistemas.

En otros entornos, el tipo de señal de contabilidad se establece en el valor ATTUNK. En estos entornos, el campo MDPAT se puede utilizar para deducir el tipo de señal de contabilidad recibida.

- Todos los demás bytes se establecen en cero binario.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETI o PMSETA en el parámetro **PMO** . Si no se especifica PMSETI ni PMSETA, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Información de contexto de control.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el MDACC que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDACC que se mantiene con el mensaje si se conserva (consulte la descripción de PMRET en “MQPMO (opciones de transferencia de mensajes) en IBM i” en la página 1200 para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDACC cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDACC en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo es totalmente binario cero.

Es un campo de salida para la llamada MQGET.

Este campo no está sujeto a ninguna conversión basada en el juego de caracteres del gestor de colas; el campo se trata como una serie de bits y no como una serie de caracteres.

El gestor de colas no hace nada con la información de este campo. La aplicación debe interpretar la información si desea utilizarla con fines contables.

Se puede utilizar el siguiente valor especial para el campo *MDACC* :

ACNONE
No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

La longitud de este campo la proporciona LNACCT. El valor inicial de este campo es ACNONE.

MDAID (serie de caracteres de 32 bytes)

Datos de la aplicación relacionados con la identidad.

Forma parte del *contexto de identidad* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

MDAID es información definida por la suite de aplicaciones y se puede utilizar para proporcionar información adicional sobre el mensaje o su originador. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETI o PMSETA en el parámetro **PMO** . Si hay un carácter nulo, el gestor de colas convierte el valor nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETI ni PMSETA, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Información de contexto de control.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el MDAID que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDAID que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDAID cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDAID en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona LNAIDD. El valor inicial de este campo es de 32 caracteres en blanco.

MDAOD (serie de caracteres de 4 bytes)

Datos de la aplicación relacionados con el origen.

Forma parte del *contexto de origen* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

MDAOD es información definida por la suite de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza.

El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETA en el parámetro **PMO** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el MDAOD que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDAOD que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDAOD cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDAOD en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo es dada por LNAORD. El valor inicial de este campo es de 4 caracteres en blanco.

MDBOC (entero con signo de 10 dígitos)

Umbral de restitución.

Es un recuento del número de veces que la llamada MQGET ha devuelto previamente el mensaje como parte de una unidad de trabajo y, posteriormente, se ha restituido. Se proporciona como ayuda a la aplicación para detectar errores de proceso que se basan en el contenido del mensaje. El recuento excluye las llamadas MQGET que han especificado cualquiera de las opciones GMBRW*.

La precisión de este recuento se ve afectada por el atributo de cola **HardenGetBackout** ; consulte “Atributos para colas” en la página 1402.

Es un campo de salida para la llamada MQGET. Se ignora para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es 0.

MDCID (serie de bits de 24 bytes)

Identificador de correlación.

Es una serie de bytes que la aplicación puede utilizar para relacionar un mensaje con otro, o para relacionar el mensaje con otro trabajo que la aplicación está realizando. El identificador de correlación es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de correlación es una serie de bytes y no una serie de caracteres, el identificador de correlación no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , la aplicación puede especificar cualquier valor. El gestor de colas transmite este valor con el mensaje y lo entrega a la aplicación que emite la solicitud get para el mensaje.

Si la aplicación especifica PMNCID, el gestor de colas genera un identificador de correlación exclusivo que se envía con el mensaje y también se devuelve a la aplicación emisora en la salida de la llamada MQPUT o MQPUT1 .

Este identificador de correlación generado se mantiene con el mensaje si se conserva y se utiliza como identificador de correlación cuando el mensaje se envía como publicación a los suscriptores que especifican CINONE en el campo SDCID en el MQSD pasado en la llamada MQSUB.

Consulte “MQPMO (opciones de transferencia de mensajes) en IBM i” en la página 1200 para obtener más detalles sobre las publicaciones retenidas

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo MDCID de la forma especificada por el campo MDREP del mensaje original, ya sea ROCMTC o ROPCI. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, MDCID es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje determinado para recuperarlo de la cola. Consulte la descripción del campo MDMID para obtener detalles sobre cómo especificar valores para este campo.

Especificar CINONE como identificador de correlación tiene el mismo efecto que no especificar MOCORI, es decir, cualquier identificador de correlación coincidirá.

Si se especifica la opción GMMUC en el parámetro **GMO** de la llamada MQGET, este campo se ignora.

Al volver de una llamada MQGET, el campo MDCID se establece en el identificador de correlación del mensaje devuelto (si lo hay).

Se pueden utilizar los siguientes valores especiales:

CINONA
No se ha especificado ningún identificador de correlación.

El valor es cero binario para la longitud del campo.

CINEWS
El mensaje es el inicio de una nueva sesión.

Este valor es reconocido por CICS bridge como indica el inicio de una nueva sesión, es decir, el inicio de una nueva secuencia de mensajes.

Para la llamada MQGET, es un campo de entrada/salida. Para las llamadas MQPUT y MQPUT1 , es un campo de entrada si no se especifica PMNCID y un campo de salida si se especifica PMNCID. La longitud de este campo la proporciona LNCID. El valor inicial de este campo es CINONE.

MDCSI (entero con signo de 10 dígitos)

Especifica el identificador de juego de caracteres de los datos de caracteres del mensaje.

Nota: Los datos de tipo carácter en MQMD y las otras estructuras de datos de IBM MQ que son parámetros en llamadas deben estar en el juego de caracteres del gestor de colas. Esto lo define el atributo **CodedCharSetId** del gestor de colas; consulte "Atributos del gestor de colas en IBM i" en la página 1434 para obtener detalles de este atributo.

Se pueden utilizar los siguientes valores especiales:

CSQM
Identificador de juego de caracteres del gestor de colas.

Los datos de caracteres del mensaje están en el juego de caracteres del gestor de colas.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia este valor en el MQMD enviado con el mensaje al identificador de juego de caracteres verdadero del gestor de colas. Como resultado, la llamada MQGET nunca devuelve el valor CSQM.

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres del mensaje están en el mismo juego de caracteres que esta estructura; este es el juego de caracteres del gestor de colas. (Sólo para MQMD, CSINHT tiene el mismo significado que CSQM).

El gestor de colas cambia este valor en el MQMD enviado con el mensaje al identificador de juego de caracteres real de MQMD. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo MDPAT en MQMD es ATBRKR.

CSEMBD
Identificador de juego de caracteres incorporado.

Los datos de tipo carácter del mensaje están en un juego de caracteres con el identificador contenido en los propios datos del mensaje. Puede haber cualquier número de identificadores de juego de caracteres incluidos en los datos del mensaje, que se aplican a diferentes partes de los datos. Este valor debe utilizarse para los mensajes PCF que contienen datos en una combinación de juegos de caracteres. Los mensajes PCF tienen un nombre de formato de FMPCF.

Especifique este valor sólo en las llamadas MQPUT y MQPUT1 . Si se especifica en la llamada MQGET, impide la conversión del mensaje.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia los valores CSQM y CSINHT en el MQMD enviado con el mensaje como se ha descrito anteriormente, pero no cambia el MQMD especificado en la llamada MQPUT o MQPUT1 . No se realiza ninguna otra comprobación en el valor especificado.

Las aplicaciones que recuperan mensajes deben comparar este campo con el valor que espera la aplicación; si los valores difieren, es posible que la aplicación necesite convertir datos de caracteres en el mensaje.

Si se especifica la opción GMCONV en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es el identificador de juego de caracteres codificado al que deben convertirse los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica (excepto que el valor CSQM o CSINHT se convierte al valor real). Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa el identificador de juego de caracteres codificado del mensaje no convertido que se devuelve a la aplicación.

De lo contrario, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es CSQM.

MDENC (entero con signo de 10 dígitos)

Codificación numérica de datos de mensaje.

Especifica la codificación numérica de los datos numéricos del mensaje; no se aplica a los datos numéricos de la propia estructura MQMD. La codificación numérica define la representación utilizada para enteros binarios, enteros decimales empaquetados y números de coma flotante.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Se define el siguiente valor especial:

ENNAT
Codificación de máquina nativa.

La codificación es el valor predeterminado para el lenguaje de programación y la máquina en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del lenguaje de programación y del entorno. Por este motivo, las aplicaciones deben compilarse utilizando los archivos de cabecera, macro, COPY o INCLUDE adecuados para el entorno en el que se ejecutará la aplicación.

Las aplicaciones que colocan mensajes normalmente deben especificar ENNAT. Las aplicaciones que recuperan mensajes deben comparar este campo con el valor ENNAT; si los valores difieren, es posible que la aplicación tenga que convertir datos numéricos en el mensaje. La opción GMCONV se puede utilizar para solicitar al gestor de colas que convierta el mensaje como parte del proceso de la llamada MQGET.

Si se especifica la opción GMCONV en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es la codificación a la que se deben convertir los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica. Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa la codificación del mensaje no convertido que se devuelve a la aplicación.

En otros casos, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es ENNAT.

MDEXP (entero con signo de 10 dígitos)

Duración del mensaje.

Es un periodo de tiempo expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. El mensaje se debe poder seleccionar para descartarlo si no se ha suprimido de la cola de destino antes de que transcurra este período de tiempo.

El valor disminuye para reflejar el tiempo que el mensaje pasa en la cola de destino, y también en cualquier cola de transmisión intermedia si la transferencia es a una cola remota. También pueden disminuir los agentes de canal de mensajes para reflejar los tiempos de transmisión, si estos son significativos. Del mismo modo, una aplicación que reenvía este mensaje a otra cola puede disminuir el valor si es necesario, si ha retenido el mensaje durante un tiempo significativo. Sin embargo, la hora de caducidad se trata como aproximada y no es necesario reducir el valor para reflejar intervalos de tiempo pequeños.

Cuando una aplicación recupera el mensaje utilizando la llamada MQGET, el campo MDEXP representa la cantidad de tiempo de caducidad original que todavía permanece.

Una vez transcurrido el tiempo de caducidad de un mensaje, éste pasa a ser elegible para ser descartado por el gestor de colas. En las implementaciones actuales, el mensaje se descarta cuando se produce una llamada MQGET de examinar o no examinar que habría devuelto el mensaje si no hubiera caducado. Por ejemplo, una llamada MQGET sin examinar con el campo GMMO en MQGMO establecido en MONONE leyendo de una cola ordenada FIFO hará que todos los mensajes caducados se descarten hasta el primer mensaje no caducado. Con una cola ordenada de prioridad, la misma llamada descartará los mensajes caducados de prioridad más alta y los mensajes de prioridad igual que han llegado a la cola antes del primer mensaje no caducado.

Un mensaje que ha caducado nunca se devuelve a una aplicación (ya sea mediante una llamada MQGET de examinar o no examinar), por lo que el valor del campo MDEXP del descriptor de mensaje después de una llamada MQGET satisfactoria es mayor que cero o el valor especial EIULIM.

Si un mensaje se coloca en una cola remota, el mensaje puede caducar (y descartarse) mientras está en una cola de transmisión intermedia, antes de que el mensaje llegue a la cola de destino.

Se genera un informe cuando se descarta un mensaje caducado, si el mensaje ha especificado una de las opciones de informe ROEXP*. Si no se especifica ninguna de estas opciones, no se genera ningún informe de este tipo; se presupone que el mensaje ya no es relevante después de este periodo de tiempo (quizás porque un mensaje posterior lo ha reemplazado).

Cualquier otro programa que descarte mensajes basándose en la hora de caducidad también debe enviar un mensaje de informe adecuado si se ha solicitado uno.

Nota:

1. Si se coloca un mensaje con una hora MDEXP de cero, la llamada MQPUT o MQPUT1 falla con el código de razón RC2013; no se genera ningún mensaje de informe en este caso.
2. Puesto que un mensaje con un tiempo de caducidad que ha transcurrido puede que no se descarte realmente hasta más adelante, puede haber mensajes en una cola que hayan pasado su tiempo de caducidad y que, por lo tanto, no sean aptos para la recuperación. Sin embargo, estos mensajes cuentan para el número de mensajes en la cola para todos los fines, incluido el desencadenamiento de profundidad.
3. Se genera un informe de caducidad, si se solicita, cuando el mensaje se descarta realmente, no cuando pasa a ser elegible para descartarlo.
4. El descarte de un mensaje caducado y la generación de un informe de caducidad si se solicita, nunca forman parte de la unidad de trabajo de la aplicación, incluso si el mensaje se ha planificado para descartarlo como resultado de una llamada MQGET que opera dentro de una unidad de trabajo.
5. Si una llamada MQGET recupera un mensaje casi caducado dentro de una unidad de trabajo, y posteriormente se restituye la unidad de trabajo, el mensaje puede ser apto para ser descartado antes de que se pueda recuperar de nuevo.
6. Si un mensaje casi caducado está bloqueado por una llamada MQGET con GMLK, el mensaje puede ser elegible para ser descartado antes de que pueda ser recuperado por una llamada MQGET con GMMUC; el código de razón RC2034 se devuelve en esta llamada MQGET posterior si esto sucede.
7. Cuando se recupera un mensaje de solicitud con un tiempo de caducidad mayor que cero, la aplicación puede realizar una de las acciones siguientes cuando envía el mensaje de respuesta:
   - Copie el tiempo de caducidad restante del mensaje de solicitud en el mensaje de respuesta.
   - Establezca la hora de caducidad en el mensaje de respuesta en un valor explícito mayor que cero.
   - Establezca la hora de caducidad en el mensaje de respuesta en EIULIM.

   La acción a realizar depende del diseño de la suite de aplicaciones. Sin embargo, la acción predeterminada para transferir mensajes a una cola de mensajes no entregados (undelivered-message) debe ser conservar el tiempo de caducidad restante del mensaje y continuar decrementándolo.
8. Los mensajes desencadenantes siempre se generan con EIULIM.
9. Un mensaje (normalmente en una cola de transmisión) que tiene un nombre MDFMT de FMXQH tiene un segundo descriptor de mensaje dentro de MQXQH. Por lo tanto, tiene dos campos MDEXP asociados. En este caso deben señalarse los siguientes puntos adicionales:
   - Cuando una aplicación coloca un mensaje en una cola remota, el gestor de colas coloca el mensaje inicialmente en una cola de transmisión local y añade un prefijo a los datos del mensaje de aplicación con una estructura MQXQH. El gestor de colas establece los valores de los dos campos MDEXP para que sean los mismos que los especificados por la aplicación.

     Si una aplicación coloca un mensaje directamente en una cola de transmisión local, los datos del mensaje ya deben empezar con una estructura MQXQH, y el nombre de formato debe ser FMXQH (pero el gestor de colas no lo impone). En este caso, no es necesario que la aplicación establezca los valores de estos dos campos MDEXP para que sean iguales. (El gestor de colas no comprueba que el campo MDEXP de MQXQH contenga un valor válido, o incluso que los datos del mensaje sean lo suficientemente largos para incluirlo.)

- Cuando se recupera un mensaje con un nombre MDFMT de FMXQH de una cola (tanto si se trata de una cola normal como de una cola de transmisión), el gestor de colas disminuye estos dos campos MDEXP con el tiempo empleado en esperar en la cola. No se genera ningún error si los datos del mensaje no son lo suficientemente largos para incluir el campo MDEXP en MQXQH.
- El gestor de colas utiliza el campo MDEXP en el descriptor de mensaje independiente (es decir, no el del descriptor de mensaje incorporado en la estructura MQXQH) para probar si el mensaje es apto para descartarlo.
- Si los valores iniciales de los dos campos MDEXP eran diferentes, por lo tanto, es posible que la hora MDEXP en el descriptor de mensaje separado cuando se recupera el mensaje sea mayor que cero (por lo que el mensaje no es elegible para descartarlo), mientras que ha transcurrido el tiempo de acuerdo con el campo MDEXP en MQXQH. En este caso, el campo MDEXP en MQXQH se establece en cero.

Se reconoce el siguiente valor especial:

EIULIM
Duración ilimitada.

El mensaje tiene un tiempo de caducidad ilimitado.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es EIULIM.

MDFB (entero con signo de 10 dígitos)

Código de retorno o de razón.

Se utiliza con un mensaje de tipo MTRPRT para indicar la naturaleza del informe y sólo es significativo con ese tipo de mensaje. El campo puede contener uno de los valores FB* o uno de los valores RC*. Los códigos de comentarios se agrupan de la forma siguiente:

FBNONE
No se han proporcionado comentarios.

FBSFST
Valor más bajo para los comentarios generados por el sistema.

FBSLST
Valor más alto para comentarios generados por el sistema.

El rango de códigos de realimentación generados por el sistema FBSFST a FBSLST incluye los códigos de realimentación generales listados más adelante en esta sección (FB*), y también los códigos de razón (RC*) que se pueden producir cuando el mensaje no se puede colocar en la cola de destino.

FBAFST
Valor más bajo para los comentarios generados por la aplicación.

FBALST
Valor más alto para los comentarios generados por la aplicación.

Las aplicaciones que generan mensajes de informe no deben utilizar códigos de comentarios en el rango del sistema (que no sean FBQUIT), a menos que deseen simular mensajes de informe generados por el gestor de colas o el agente de canal de mensajes.

En las llamadas MQPUT o MQPUT1 , el valor especificado debe ser FBNONE o estar dentro del rango del sistema o del rango de aplicaciones. Esto se comprueba independientemente del valor de MDMT.

Códigos de comentarios generales:

FBCOA
Confirmación de llegada a la cola de destino (consulte ROCOA).

FBCOD
Confirmación de la entrega a la aplicación receptora (ver ROCOD).

FBEXP
El mensaje ha caducado.

El mensaje se ha descartado porque no se había eliminado de la cola de destino antes de que hubiera transcurrido su tiempo de caducidad.

FBPAN
Notificación de acción positiva (consulte ROPAN).

FBNAN
Notificación de acción negativa (véase RONAN).

FBQUIT
La aplicación debe finalizar.

Esto lo puede utilizar un programa de planificación de carga de trabajo para controlar el número de instancias de un programa de aplicación que se están ejecutando. El envío de un mensaje MTRPRT con este código de retorno a una instancia del programa de aplicación indica a dicha instancia que debe detener el proceso. Sin embargo, el cumplimiento de este convenio es un asunto de la aplicación; el gestor de colas no lo impone.

IMS-bridge feedback codes: cuando el puente IMS recibe un código de detección IMS-OTMA distinto de cero, el puente IMS convierte el código de detección de hexadecimal a decimal, añade el valor FBIERR (300) y coloca el resultado en el campo MDFB del mensaje de respuesta. Esto hace que el código de retorno tenga un valor en el rango de FBIFST (301) a FBILST (399) cuando se ha producido un error IMS-OTMA.

El puente IMS puede generar los siguientes códigos de comentarios:

FBDLZ
Longitud de datos cero.

Una longitud de segmento era cero en los datos de aplicación del mensaje.

FBDLN
Longitud de datos negativa.

Una longitud de segmento era negativa en los datos de aplicación del mensaje.

FBDLTB
Longitud de datos demasiado grande.

Una longitud de segmento era demasiado grande en los datos de aplicación del mensaje.

FBBUFO
Desbordamiento de almacenamiento intermedio.

El valor de uno de los campos de longitud haría que los datos desbordaran el almacenamiento intermedio de mensajes.

FBLOB1
Longitud errónea por uno.

El valor de uno de los campos de longitud era un byte demasiado corto.

FBIH
La estructura MQIIH no es válida o falta.

El campo MDFMT en MQMD especifica FMIMS, pero el mensaje no empieza por una estructura MQIIH válida.

FBNAFI
ID de usuario no autorizado para su uso en IMS.

El ID de usuario contenido en el descriptor de mensaje MQMD, o la contraseña contenida en el campo IIAUT de la estructura MQIIH, ha fallado la validación realizada por el puente IMS . Como resultado, el mensaje no se ha pasado a IMS.

FBIERR
Error inesperado devuelto por IMS.

IMSha devuelto un error inesperado. Consulte el registro de errores de IBM MQ en el sistema en el que reside el puente IMS para obtener más información sobre el error.

FBIFST
Valor más bajo para los comentarios generados por IMS.

Los códigos de retorno generados por IMSocupan el rango de FBIFST (300) a FBILST (399). El propio código de detección IMS-OTMA es MDFB menos FBIERR.

FBILST
Valor más alto para los comentarios generados por IMS.

CICS-bridge feedback codes: CICS bridgepuede generar los siguientes códigos de comentarios:

FBCAAB
La aplicación ha terminado de forma anómala.

El programa de aplicación especificado en el mensaje ha terminado de forma anómala. Este código de retorno sólo aparece en el campo DLREA de la estructura MQDLH.

FBCANS
No se puede iniciar la aplicación.

El EXEC CICS LINK para el programa de aplicación especificado en el mensaje ha fallado. Este código de retorno sólo aparece en el campo DLREA de la estructura MQDLH.

FBCBRF
CICS bridge ha terminado de forma anómala sin completar el proceso de errores normal.

FBCCSE
Identificador de juego de caracteres no válido.

FBCIHE
Falta la estructura de cabecera de información de CICS o no es válida.

FBCCAA
La longitud del área de comunicación de CICS no es válida.

FBCCIE
Identificador de correlación no válido.

FBCDLQ
Cola de mensajes no entregados no disponible.

La tarea CICS bridge no ha podido copiar una respuesta a esta solicitud en la cola de mensajes no entregados. La solicitud se ha restituido.

FBCENO
Codificación no válida.

FBCINE
CICS bridge ha encontrado un error inesperado.

Este código de retorno sólo aparece en el campo DLREA de la estructura MQDLH.

FBCNTA
Identificador de usuario no autorizado o contraseña no válida.

Este código de retorno sólo aparece en el campo DLREA de la estructura MQDLH.

FBCUBO
Unidad de trabajo restituido.

La unidad de trabajo fue restituido, por una de las siguientes razones:

- Se ha detectado una anomalía al procesar otra solicitud dentro de la misma unidad de trabajo.

- Se ha producido una terminación anómala de CICS mientras la unidad de trabajo estaba en curso.

FBCUWE
Campo de control de unidad de trabajo CIUOW no válido.

Códigos de razón deMQ: para mensajes de informe de excepción, MDFB contiene un código de razón de MQ . Entre los posibles códigos de razón están:

RC2051
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

RC2053
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

RC2035
(2035, X'7F3') No autorizado para el acceso.

RC2056
(2056, X'808 ') No hay espacio disponible en disco para la cola.

RC2048
(2048, X'800 ') La cola no admite mensajes persistentes.

RC2031
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

RC2030
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es FBNONE.

MDFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos del mensaje.

Es un nombre que el remitente del mensaje puede utilizar para indicar al destinatario la naturaleza de los datos del mensaje. Se puede especificar cualquier carácter que esté en el juego de caracteres del gestor de colas para el nombre, pero se recomienda que el nombre esté restringido a lo siguiente:

- Mayúsculas de la A a la Z
- Dígitos numéricos del 0 al 9

Si se utilizan otros caracteres, puede que no sea posible convertir el nombre entre los juegos de caracteres de los gestores de colas emisor y receptor.

El nombre debe rellenarse con espacios en blanco hasta la longitud del campo, o un carácter nulo utilizado para terminar el nombre antes del final del campo; el nulo y los caracteres subsiguientes se tratan como blancos. No especifique un nombre con espacios en blanco iniciales o intercalados. Para la llamada MQGET, el gestor de colas devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

El gestor de colas no comprueba que el nombre cumple con las recomendaciones descritas anteriormente.

Los nombres que empiezan por "MQ" en mayúsculas, minúsculas y combinación de mayúsculas y minúsculas tienen significados definidos por el gestor de colas; no debe utilizar nombres que empiecen por estas letras para sus propios formatos. Los formatos incorporados del gestor de colas son:

FMNONE
No hay nombre de formato.

La naturaleza de los datos no está definida. Esto significa que los datos no se pueden convertir cuando el mensaje se recupera de una cola utilizando la opción GMCONV.

Si se especifica GMCONV en la llamada MQGET, y el juego de caracteres o codificación de datos del mensaje difiere del especificado en el parámetro **MSGDSC** , el mensaje se devuelve con los siguientes códigos de terminación y de razón (suponiendo que no haya otros errores):

- Código de terminación CCWARN y código de razón RC2110 si los datos FMNONE están al principio del mensaje.
- Código de terminación CCOK y código de razón RCNONE si los datos FMNONE están al final del mensaje (es decir, precedidos por una o más estructuras de cabecera MQ). Las estructuras de cabecera MQ se convierten al juego de caracteres solicitado y a la codificación en este caso.

FMADMN
Mensaje de solicitud/respuesta del servidor de mandatos.

El mensaje es un mensaje de petición o respuesta de servidor de mandatos en formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET. Para obtener más información sobre cómo utilizar mensajes de formato de mandato programable, consulte Utilización de formatos de mandato programable.

FMCICS
Cabecera de información de CICS .

Los datos del mensaje empiezan con la cabecera de información MQCIH de CICS , que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo CIFMT en la estructura MQCIH.

FMCMD1
Mensaje de respuesta de mandato de tipo 1.

El mensaje es un mensaje de respuesta de servidor de mandatos MQSC que contiene el recuento de objetos, el código de terminación y el código de razón. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMCMD2
Mensaje de respuesta de mandato de tipo 2.

El mensaje es un mensaje de respuesta de servidor de mandatos MQSC que contiene información sobre los objetos solicitados. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMDLH
Cabecera de mensaje no entregado.

Los datos del mensaje empiezan por la cabecera MQDLH de mensaje no entregado. Los datos del mensaje original siguen inmediatamente la estructura MQDLH. El nombre de formato de los datos de mensaje originales lo proporciona el campo DLFMT en la estructura MQDLH; consulte “MQDLH (cabecera Dead-letter) en IBM i” en la página 1086 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

Los informes COA y COD no se generan para los mensajes que tienen un MDFMT de FMDLH.

FMDH
Cabecera de lista de distribución.

Los datos del mensaje empiezan con la cabecera de lista de distribución MQDH; esto incluye las matrices de registros MQOR y MQPMR. La cabecera de lista de distribución puede ir seguida de datos adicionales. El formato de los datos adicionales (si los hay) se proporciona mediante el campo DHFMT en la estructura MQDH; consulte “MQDH (Cabecera de distribución) en IBM i” en la página 1081 para obtener detalles de esta estructura. Los mensajes con formato FMDH se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMEVNT
Mensaje de suceso.

El mensaje es un mensaje de suceso de MQ que notifica un suceso que se ha producido. Los mensajes de suceso tienen la misma estructura que los mandatos programables; para obtener más información sobre esta estructura, consulte Estructuras para mandatos y respuestas. Para obtener información sobre sucesos, consulte Supervisión de sucesos.

Los mensajes de suceso Version-1 se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMIMS
Cabecera de información de IMS .

Los datos del mensaje empiezan con la cabecera de información MQIIH de IMS , que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo *IIFMT* en la estructura MQIIH. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMIMVS
Serie de variable IMS .

El mensaje es una serie de variable IMS , que es una serie con el formato `llzzccc`, donde:

`ll`
es un campo de longitud de 2 bytes que especifica la longitud total del elemento de serie de variable IMS . Esta longitud es igual a la longitud de `ll` (2 bytes), más la longitud de `zz` (2 bytes), más la longitud de la propia serie de caracteres. `ll` es un entero binario de 2 bytes en la codificación especificada por el campo `MDENC` .

`zz`
es un campo de 2 bytes que contiene distintivos que son significativos para IMS. `zz` es una serie de bytes que consta de dos campos de serie de bits de 1 byte y se transmite sin cambiar de remitente a destinatario (es decir, `zz` no está sujeto a ninguna conversión).

`ccc`
es una serie de caracteres de longitud variable que contiene `ll-4` caracteres. `ccc` está en el juego de caracteres especificado por el campo `MDCSI` .

Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMMDE
Extensión de descriptor de mensaje.

Los datos del mensaje empiezan con la extensión de descriptor de mensaje MQMDE y, opcionalmente, van seguidos de otros datos (normalmente, los datos del mensaje de aplicación). El nombre de formato, el juego de caracteres y la codificación de los datos que siguen a MQMDE se proporcionan mediante los campos `MEFMT`, `MECSI`y `MEENC` en MQMDE. Consulte “MQMDE (extensión de descriptor de mensaje) en IBM i” en la página 1179 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMPCF
Mensaje definido por el usuario en formato de mandato programable (PCF).

El mensaje es un mensaje definido por el usuario que se ajusta a la estructura de un mensaje de formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET. Consulte Utilización de formatos de mandatos programables para obtener más información sobre la utilización de mensajes de formato de mandatos programables.

HMRM
Cabecera de mensaje de referencia.

Los datos del mensaje empiezan con la cabecera de mensaje de referencia MQRMH y, opcionalmente, van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos se proporcionan mediante los campos `RMFMT`, `RMCSI`y `RMENC` en

MQRMH. Consulte “MQRMH (cabecera de mensaje de referencia) en IBM i” en la página 1228 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMRFH
Reglas y cabecera de formato.

Los datos del mensaje empiezan con las reglas y la cabecera de formato MQRFH, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos (si los hay) los proporcionan los campos RFFMT, RFCSIy RFENC en MQRFH. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMRFH2
Reglas y formato de cabecera versión 2.

Los datos del mensaje empiezan con las reglas version-2 y la cabecera de formato MQRFH2, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos opcionales (si los hay) los proporcionan los campos RF2FMT, RF2CSIy RF2ENC en MQRFH2. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMSTR
Mensaje que consta totalmente de caracteres.

Los datos del mensaje de aplicación pueden ser una serie SBCS (juego de caracteres de un solo byte) o una serie DBCS (juego de caracteres de doble byte). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMTM
Mensaje desencadenante.

El mensaje es un mensaje desencadenante, descrito por la estructura MQTM; consulte “MQTM-Mensaje de desencadenante” en la página 1264 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMWIH
Cabecera de información de trabajo.

Los datos del mensaje empiezan con la cabecera de información de trabajo MQWIH, que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo WIFMT en la estructura MQWIH.

FMXQH
Cabecera de cola de transmisión.

Los datos del mensaje empiezan por la cabecera de cola de transmisión MQXQH. Los datos del mensaje original siguen inmediatamente a la estructura MQXQH. El nombre de formato de los datos de mensaje originales lo proporciona el campo MDFMT de la estructura MQMD que forma parte de la cabecera de cola de transmisión MQXQH. Consulte “MQXQH (cabecera de cola de transmisión) en IBM i” en la página 1274 para obtener detalles de esta estructura.

Los informes COA y COD no se generan para los mensajes que tienen un MDFMT de FMXQH.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

MDGID (serie de bits de 24 bytes)

Identificador de grupo.

Es una serie de bytes que se utiliza para identificar el grupo de mensajes o el mensaje lógico al que pertenece el mensaje físico. MDGID también se utiliza si se permite la segmentación para el mensaje. En todos estos casos, MDGID tiene un valor no nulo y uno o varios de los distintivos siguientes se establecen en el campo MDMFL :

- MFMIG
- MFLMIG
- MFSEG
- MFLSEG
- MFSEGA

Si no se establece ninguno de estos distintivos, MDGID tiene el valor nulo especial GINONE.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se ha especificado MOGRPI.

Considere la posibilidad de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que MDGID esté establecido en un valor adecuado.

Los grupos de mensajes y segmentos sólo se pueden procesar correctamente si el identificador de grupo es exclusivo. Por esta razón, las aplicaciones no deben generar sus propios identificadores de grupo; en su lugar, las aplicaciones deben realizar una de las acciones siguientes:

- Si se especifica PMLOGO, el gestor de colas genera automáticamente un identificador de grupo exclusivo para el primer mensaje del grupo o segmento del mensaje lógico, y utiliza ese identificador de grupo para los mensajes restantes del grupo o segmentos del mensaje lógico, por lo que la aplicación no necesita realizar ninguna acción especial. Considere la posibilidad de utilizar este procedimiento.
- Si no se especifica PMLOGO, la aplicación debe solicitar al gestor de colas que genere el identificador de grupo, estableciendo MDGID en GINONE en la primera llamada MQPUT o MQPUT1 para un mensaje del grupo o segmento del mensaje lógico. El identificador de grupo devuelto por el gestor de colas en la salida de esa llamada debe utilizarse entonces para los mensajes restantes en el grupo o segmentos del mensaje lógico. Si un grupo de mensajes contiene mensajes segmentados, se debe utilizar el mismo identificador de grupo para todos los segmentos y mensajes del grupo.

  Cuando no se especifica PMLOGO, los mensajes de grupos y segmentos de mensajes lógicos pueden colocarse en cualquier orden (por ejemplo, en orden inverso), pero el identificador de grupo debe ser asignado por la primera llamada MQPUT o MQPUT1 que se emite para cualquiera de estos mensajes.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor detallado en PMOPT. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje si el objeto abierto es una sola cola y no una lista de distribución, pero lo deja sin modificar si el objeto abierto es una lista de distribución. En este último caso, si la aplicación necesita conocer los identificadores de grupo generados, la aplicación debe proporcionar registros MQPMR que contengan el campo PRGID .

En la entrada a la llamada MQGET, el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

Se define el siguiente valor especial:

GINONA
No se ha especificado ningún identificador de grupo.

El valor es cero binario para la longitud del campo. Es el valor que se utiliza para los mensajes que no están en grupos, no en segmentos de mensajes lógicos y para los que no se permite la segmentación.

La longitud de este campo la proporciona LNGID. El valor inicial de este campo es GINONE. Este campo se ignora si MDVER es menor que MDVER2.

MDMFL (entero con signo de 10 dígitos)

Distintivos de mensaje.

Son distintivos que especifican atributos del mensaje o controlan su proceso. Los distintivos se dividen en las categorías siguientes:

- Distintivo de segmentación
- Distintivos de estado

Estos se describen a su vez.

Distintivos de segmentación: cuando un mensaje es demasiado grande para una cola, normalmente falla un intento de colocar el mensaje en la cola. La segmentación es una técnica mediante la cual el gestor de colas o la aplicación divide el mensaje en partes más pequeñas denominadas segmentos, y coloca cada segmento en la cola como un mensaje físico independiente. La aplicación que recupera el mensaje puede recuperar los segmentos uno por uno o solicitar al gestor de colas que vuelva a ensamblar los segmentos en un único mensaje devuelto por la llamada MQGET. Esto último se consigue especificando la opción GMCMPM en la llamada MQGET y proporcionando un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo. (Consulte "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener detalles de la opción GMCMPM.) La segmentación de un mensaje puede producirse en el gestor de colas emisor, en un gestor de colas intermedio o en el gestor de colas de destino.

Puede especificar una de las siguientes opciones para controlar la segmentación de un mensaje:

MFSEGI
Segmentación inhibida.

Esta opción impide que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción impide que el segmento se divida en segmentos más pequeños.

El valor de este distintivo es cero binario. Éste es el valor predeterminado.

MFSEGA
Segmentación permitida.

Esta opción permite que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción permite que el segmento se divida en segmentos más pequeños. Se puede establecer MFSEGA sin establecer MFSEG o MFLSEG.

Cuando el gestor de colas segmenta un mensaje, el gestor de colas activa el distintivo MFSEG en la copia del MQMD que se envía con cada segmento, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 . Para el último segmento del mensaje lógico, el gestor de colas también activa el distintivo MFLSEG en el MQMD que se envía con el segmento.

Nota: Es necesario tener cuidado cuando los mensajes se colocan con MFSEGA pero sin PMLOGO. Si el mensaje es:

- No es un segmento, y
- No en un grupo, y
- No se reenvía,

la aplicación debe recordar restablecer el campo `MDGID` en GINONE antes de cada llamada MQPUT o MQPUT1 , para que el gestor de colas genere un identificador de grupo exclusivo para cada mensaje. Si esto no se hace, los mensajes no relacionados podrían terminar inadvertidamente con el mismo identificador de grupo, lo que podría llevar a un proceso incorrecto posteriormente. Consulte las descripciones del campo `MDGID` y la opción PMLOGO para obtener más información sobre cuándo se debe restablecer el campo `MDGID` .

El gestor de colas divide los mensajes en segmentos según sea necesario para asegurarse de que los segmentos (más los datos de cabecera que puedan ser necesarios) caben en la cola. Sin embargo, hay un límite inferior para el tamaño de un segmento generado por el gestor de colas, y

sólo el último segmento creado a partir de un mensaje puede ser menor que este límite. (El límite inferior para el tamaño de un segmento generado por la aplicación es de un byte.) Los segmentos generados por el gestor de colas pueden tener una longitud desigual. El gestor de colas procesa el mensaje como se indica a continuación:

- Los formatos definidos por el usuario se dividen en límites que son múltiplos de 16 bytes. Esto significa que el gestor de colas no generará segmentos que sean menores de 16 bytes (que no sean el último segmento).
- Los formatos incorporados distintos de FMSTR se dividen en puntos adecuados a la naturaleza de los datos presentes. Sin embargo, el gestor de colas nunca divide un mensaje en medio de una estructura de cabecera de MQ . Esto significa que el gestor de colas no puede dividir más un segmento que contenga una única estructura de cabecera MQ y, como resultado, el tamaño de segmento mínimo posible para ese mensaje es mayor que 16 bytes.

  El segundo segmento o segmento posterior generado por el gestor de colas empezará por uno de los siguientes:

  – Una estructura de cabecera MQ
  – El inicio de los datos del mensaje de aplicación
  – Parte a través de los datos del mensaje de aplicación
- FMSTR se divide sin tener en cuenta la naturaleza de los datos presentes (SBCS, DBCS o SBCS/DBCS mixto). Cuando la serie es DBCS o SBCS/DBCS mixto, esto puede dar como resultado segmentos que no se pueden convertir de un juego de caracteres a otro. El gestor de colas nunca divide los mensajes FMSTR en segmentos menores de 16 bytes (que no sean el último segmento).
- El gestor de colas establece los campos `MDFMT`, `MDCSI`y `MDENC` en el MQMD de cada segmento para describir correctamente los datos presentes al principio del segmento; el nombre de formato será el nombre de un formato incorporado o el nombre de un formato definido por el usuario.
- El campo `MDREP` en el MQMD de segmentos con `MDOFF` mayor que cero se modifica como se indica a continuación:

  – Para cada tipo de informe, si la opción de informe es RO* D, pero el segmento no puede contener posiblemente ninguno de los primeros 100 bytes de datos de usuario (es decir, los datos que siguen a las estructuras de cabecera de MQ que pueden estar presentes), la opción de informe se cambia a RO*.

El gestor de colas sigue las reglas anteriores, pero de lo contrario divide los mensajes de forma imprevisible; no realice suposiciones sobre dónde se divide un mensaje

Para los mensajes persistentes, el gestor de colas sólo puede realizar la segmentación dentro de una unidad de trabajo:

- Si la llamada MQPUT o MQPUT1 está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de segmentación, el gestor de colas elimina los segmentos que se colocaron en la cola como resultado de la llamada anómala. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo sólo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas restituye la unidad de trabajo.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar la segmentación. Si el mensaje no requiere segmentación, la llamada puede seguir siendo satisfactoria. Pero si el mensaje no requiere segmentación, la llamada falla con el código de razón RC2255.

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para poder realizar la segmentación.

Deberá prestarse especial atención a la conversión de datos de los mensajes que puedan segmentarse:

- Si la conversión de datos sólo la realiza la aplicación receptora en la llamada MQGET y la aplicación especifica la opción GMCMPM, la salida de conversión de datos pasará el mensaje completo para que la salida se convierta y el hecho de que el mensaje se haya segmentado no será aparente para la salida.
- Si la aplicación receptora recupera un segmento a la vez, se invocará la salida de conversión de datos para convertir un segmento a la vez. Por lo tanto, la salida debe ser capaz de convertir los datos en un segmento independientemente de los datos de cualquiera de los otros segmentos.

  Si la naturaleza de los datos en el mensaje es tal que la segmentación arbitraria de los datos en límites de 16 bytes puede dar como resultado segmentos que la salida no puede convertir, o el formato es FMSTR y el juego de caracteres es DBCS o SBCS/DBCS mixto, la propia aplicación emisora debe crear y poner los segmentos, especificando MFSEGI para suprimir la segmentación adicional. De esta forma, la aplicación emisora puede asegurarse de que cada segmento contiene información suficiente para permitir que la salida de conversión de datos convierta el segmento correctamente.
- Si se especifica la conversión del emisor para un agente de canal de mensajes (MCA) de envío, el MCA sólo convierte los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir los mensajes que son segmentos.

Este distintivo es un distintivo de entrada en las llamadas MQPUT y MQPUT1 y un distintivo de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite el valor del distintivo en el campo GMSEG en MQGMO.

El valor inicial de este distintivo es MFSEGI.

Distintivos de estado: son distintivos que indican si el mensaje físico pertenece a un grupo de mensajes, es un segmento de un mensaje lógico, ambos o ninguno. Se pueden especificar uno o más de los siguientes valores en la llamada MQPUT o MQPUT1 , o la llamada MQGET los puede devolver:

MFMIG
El mensaje es miembro de un grupo.

MFLMIG
El mensaje es el último mensaje lógico de un grupo.

Si se establece este distintivo, el gestor de colas activa MFMIG en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Es válido que un grupo conste de un solo mensaje lógico. Si este es el caso, se establece MFLMIG, pero el campo MDSEQ tiene el valor uno.

MFSEG
El mensaje es un segmento de un mensaje lógico.

Cuando se especifica MFSEG sin MFLSEG, la longitud de los datos del mensaje de aplicación en el segmento (excluyendo las longitudes de las estructuras de cabecera de MQ que pueden estar presentes) debe ser como mínimo una. Si la longitud es cero, la llamada MQPUT o MQPUT1 falla con el código de razón RC2253.

MFLSEG
El mensaje es el último segmento de un mensaje lógico.

Si se establece este distintivo, el gestor de colas activa MFSEG en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Es válido que un mensaje lógico conste de un solo segmento. Si este es el caso, se establece MFLSEG, pero el campo MDOFF tiene el valor cero.

Cuando se especifica MFLSEG, se permite que la longitud de los datos del mensaje de aplicación en el segmento (excluyendo las longitudes de cualquier estructura de cabecera que pueda estar presente) sea cero.

La aplicación debe asegurarse de que estos distintivos se establecen correctamente al transferir mensajes. Si se especifica PMLOGO, o se ha especificado en la llamada MQPUT anterior para el descriptor de contexto de cola, los valores de los distintivos deben ser coherentes con la información de grupo y segmento retenida por el gestor de colas para el descriptor de contexto de cola. Las condiciones siguientes se aplican a las llamadas MQPUT sucesivas para el descriptor de contexto de cola cuando se especifica PMLOGO:

- Si no hay ningún grupo actual o mensaje lógico, todos estos distintivos (y combinaciones de ellos) son válidos.
- Una vez que se ha especificado MFMIG, debe permanecer activado hasta que se especifique MFLMIG. La llamada falla con el código de razón RC2241 si no se cumple esta condición.
- Una vez que se ha especificado MFSEG, debe permanecer activado hasta que se especifique MFLSEG. La llamada falla con el código de razón RC2242 si no se cumple esta condición.
- Una vez que se ha especificado MFSEG sin MFMIG, MFMIG debe permanecer desactivado hasta que se haya especificado MFLSEG. La llamada falla con el código de razón RC2242 si no se cumple esta condición.

La Tabla 1 muestra las combinaciones válidas de los distintivos y los valores utilizados para diversos campos.

Estos distintivos son distintivos de entrada en las llamadas MQPUT y MQPUT1 y distintivos de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite los valores de los distintivos en los campos GMGST y GMSST en MQGMO.

Distintivos predeterminados: se puede especificar lo siguiente para indicar que el mensaje tiene atributos predeterminados:

MFNONE
Sin distintivos de mensaje (atributos de mensaje predeterminados).

Esto inhibe la segmentación e indica que el mensaje no está en un grupo y no es un segmento de un mensaje lógico. MFNONE está definido para ayudar a la documentación del programa. No se pretende que este distintivo se utilice con ningún otro, pero como su valor es cero, no se puede detectar dicho uso.

El campo MDMFL se particiona en subcampos; para obtener detalles, consulte “Opciones de informe y distintivos de mensaje en IBM i” en la página 1469.

El valor inicial de este campo es MFNONE. Este campo se ignora si MDVER es menor que MDVER2.

MDMID (serie de bits de 24 bytes)

Identificador del mensaje.

Es una serie de bytes que se utiliza para distinguir un mensaje de otro. Generalmente, ningún mensaje debe tener el mismo identificador de mensaje, aunque el gestor de colas no lo permita. El identificador de mensaje es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de mensaje es una serie de bytes y no una serie de caracteres, el identificador de mensaje no se convierte entre juegos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , si la aplicación especifica MINONE o PMNMID, el gestor de colas genera un identificador de mensaje exclusivo cuando se coloca el mensaje y lo coloca en el descriptor de mensaje enviado con el mensaje. El gestor de colas también devuelve este identificador de mensaje en el descriptor de mensaje que pertenece a la aplicación emisora. La aplicación puede utilizar este valor para registrar información sobre mensajes concretos y para responder a consultas de otras partes de la aplicación.

Un MDMID generado por el gestor de colas consta de un identificador de producto de 4 bytes (AMQ¬ o CSQ¬ en ASCII o EBCDIC, donde ¬ representa un único carácter en blanco), seguido de una

implementación específica del producto de una serie exclusiva. En IBM MQ , contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para asegurarse de que los identificadores de mensaje son exclusivos. La capacidad de generar una serie exclusiva también depende de que el reloj del sistema no se cambie hacia atrás. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, la aplicación debe evitar generar identificadores con caracteres iniciales en el rango de A a I en ASCII o EBCDIC (de X'41 'a X'49' y de X'C1'a X'C9'). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales en estos rangos.

Si el mensaje se está colocando en un tema, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario para cada mensaje publicado. Si la aplicación especifica PMNMID, el gestor de colas genera un identificador de mensaje exclusivo para devolver en la salida. Si MINONE está especificado por la aplicación, el valor del campo MDMID en MQMD no se modifica cuando se devuelve de la llamada.

Consulte la descripción de PMRET en PMOPT para obtener más detalles sobre las publicaciones retenidas.

Si el mensaje se está colocando en una lista de distribución, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario, pero el valor del campo MDMID en MQMD no se modifica en el retorno de la llamada, incluso si se ha especificado MINONE o PMNMID. Si la aplicación necesita conocer los identificadores de mensaje generados por el gestor de colas, la aplicación debe proporcionar registros MQPMR que contengan el campo PRMID .

La aplicación emisora también puede especificar un valor determinado para el identificador de mensaje, que no sea MINONE; esto detiene el gestor de colas que genera un identificador de mensaje exclusivo. Una aplicación que reenvía un mensaje puede utilizar este recurso para propagar el identificador de mensaje del mensaje original.

El propio gestor de colas no hace ningún uso de este campo excepto para:

- Generar un valor exclusivo si se solicita, tal como se ha descrito anteriormente
- Entregue el valor a la aplicación que emite la solicitud de obtención para el mensaje
- Copie el valor en el campo MDCID de cualquier mensaje de informe que genere sobre este mensaje (en función de las opciones de MDREP)

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo MDMID de la forma especificada por el campo MDREP del mensaje original, ya sea RONMI o ROPMI. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, MDMID es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje determinado para recuperarlo de la cola. Normalmente, la llamada MQGET devuelve el siguiente mensaje en la cola, pero si se necesita un mensaje determinado, esto se puede obtener especificando uno o más de los cinco criterios de selección, en cualquier combinación; estos campos son:

- MDMID
- MDCID
- MDGID
- MDSEQ
- MDOFF

La aplicación establece uno o varios de estos campos en los valores necesarios y, a continuación, establece las opciones de coincidencia MO* correspondientes en el campo GMMO en MQGMO para indicar que estos campos deben utilizarse como criterios de selección. Sólo los mensajes que tienen los valores especificados en esos campos son candidatos para la recuperación. El valor predeterminado para el campo GMMO (si no lo modifica la aplicación) es que coincida con el identificador de mensaje y el identificador de correlación.

Normalmente, el mensaje devuelto es el primer mensaje de la cola que cumple los criterios de selección. Pero si se especifica GMBRWN, el mensaje devuelto es el siguiente mensaje que cumple los criterios de selección; la exploración de este mensaje empieza por el mensaje que sigue a la posición actual del cursor.

Nota: La cola se explora secuencialmente en busca de un mensaje que cumpla los criterios de selección, por lo que los tiempos de recuperación serán más lentos que si no se especifica ningún criterio de selección, especialmente si se tienen que explorar muchos mensajes antes de encontrar uno adecuado.

Consulte la Tabla 1 para obtener más información sobre cómo se utilizan los criterios de selección en diversas situaciones.

Especificar MINONE como identificador de mensaje tiene el mismo efecto que no especificar MOMSGI, es decir, cualquier identificador de mensaje coincidirá.

Este campo se ignora si se especifica la opción GMMUC en el parámetro **GMO** de la llamada MQGET.

Al volver de una llamada MQGET, el campo `MDMID` se establece en el identificador de mensaje del mensaje devuelto (si lo hay).

Se puede utilizar el siguiente valor especial:

MINONA
No se ha especificado ningún identificador de mensaje.

El valor es cero binario para la longitud del campo.

Es un campo de entrada/salida para las llamadas MQGET, MQPUT y MQPUT1 . La longitud de este campo la proporciona LNMID. El valor inicial de este campo es MINONE.

MDMT (entero con signo de 10 dígitos)

Tipo de mensaje.

Indica el tipo del mensaje. Los tipos de mensaje se agrupan de la forma siguiente:

MTSFST
Valor más bajo para tipos de mensajes definidos por el sistema.

MTSLST
Valor más alto para tipos de mensajes definidos por el sistema.

Los valores siguientes están definidos actualmente dentro del rango del sistema:

MTDGRM
El mensaje no requiere una respuesta.

El mensaje es uno que no requiere una respuesta.

MTRQST
Mensaje que requiere una respuesta.

El mensaje es uno que requiere una respuesta.

El nombre de la cola a la que debe enviarse la respuesta debe especificarse en el campo `MDRQ` . El campo `MDREP` indica cómo se van a establecer `MDMID` y `MDCID` de la respuesta.

MTRPLY
Responda a un mensaje de solicitud anterior.

El mensaje es la respuesta a un mensaje de solicitud anterior (MTRQST). El mensaje debe enviarse a la cola indicada por el campo `MDRQ` del mensaje de solicitud. El campo `MDREP` de la solicitud se debe utilizar para controlar cómo se establecen `MDMID` y `MDCID` de la respuesta.

Nota: El gestor de colas no impone la relación de solicitud-respuesta; esto es una responsabilidad de la aplicación.

MTRPRT
Mensaje de informe.

El mensaje está informando sobre alguna aparición esperada o inesperada, normalmente relacionada con algún otro mensaje (por ejemplo, se ha recibido un mensaje de solicitud que contenía datos que no eran válidos). El mensaje debe enviarse a la cola indicada por el campo MDRQ del descriptor de mensaje del mensaje original. El campo MDFB debe establecerse para indicar la naturaleza del informe. El campo MDREP del mensaje original se puede utilizar para controlar cómo se deben establecer MDMID y MDCID del mensaje de informe.

Los mensajes de informe generados por el gestor de colas o el agente de canal de mensajes siempre se envían a la cola MDRQ , con los campos MDFB y MDCID establecidos como se ha descrito anteriormente.

Otros valores dentro del rango del sistema se pueden definir en futuras versiones de la MQI y las llamadas MQPUT y MQPUT1 los aceptan sin errores.

También se pueden utilizar valores definidos por la aplicación. Deben estar dentro del rango siguiente:

MTAFST
Valor más bajo para tipos de mensajes definidos por la aplicación.

MTALST
Valor más alto para tipos de mensajes definidos por la aplicación.

Para las llamadas MQPUT y MQPUT1 , el valor MDMT debe estar dentro del rango definido por el sistema o del rango definido por la aplicación; si no lo está, la llamada falla con el código de razón RC2029.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MTDGRM.

MDOFF (entero con signo de 10 dígitos)

Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico.

Es el desplazamiento en bytes de los datos del mensaje físico desde el inicio del mensaje lógico del que forman parte los datos. Estos datos se denominan *segmento*. El desplazamiento está en el rango de 0 a 999.999.999. Un mensaje físico que no es un segmento de un mensaje lógico tiene un desplazamiento de cero.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se ha especificado MOOFFS.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación no cumple estas condiciones, o la llamada es MQPUT1, la aplicación debe asegurarse de que MDOFF esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

Para un mensaje de informe que informa sobre un segmento de un mensaje lógico, el campo MDOLN (siempre que no sea OLUNDF) se utiliza para actualizar el desplazamiento en la información de segmento retenida por el gestor de colas.

En la entrada a la llamada MQGET, el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es cero. Este campo se ignora si MDVER es menor que MDVER2.

MDOLN (entero con signo de 10 dígitos)

Longitud del mensaje original.

Este campo sólo es relevante para los mensajes de informe que son segmentos. Especifica la longitud del segmento de mensaje con el que se relaciona el mensaje de informe; no especifica la longitud del mensaje lógico del que forma parte el segmento, ni la longitud de los datos del mensaje de informe.

Nota: Al generar un mensaje de informe para un mensaje que es un segmento, el gestor de colas y el agente de canal de mensajes copian en el MQMD del mensaje de informe los campos MDGID, MDSEQ, MDOFFy *MDMFL*, del mensaje original. Como resultado, el mensaje de informe también es un segmento. Se recomienda que las aplicaciones que generan mensajes de informe hagan lo mismo y que se asegure de que el campo MDOLN se ha establecido correctamente.

Se define el siguiente valor especial:

OLUNDF
No se ha definido la longitud original del mensaje.

MDOLN es un campo de entrada en las llamadas MQPUT y MQPUT1 , pero el valor proporcionado por la aplicación sólo se acepta en determinadas circunstancias:

- Si el mensaje que se está colocando es un segmento y también es un mensaje de informe, el gestor de colas acepta el valor especificado. El valor debe ser:
  - Mayor que cero si el segmento no es el último segmento
  - No menor que cero si el segmento es el último segmento
  - No menor que la longitud de los datos presentes en el mensaje

  Si no se cumplen estas condiciones, la llamada falla con el código de razón RC2252.
- Si el mensaje que se está colocando es un segmento pero no un mensaje de informe, el gestor de colas ignora el campo y utiliza en su lugar la longitud de los datos del mensaje de aplicación.
- En todos los demás casos, el gestor de colas ignora el campo y utiliza el valor OLUNDF en su lugar.

Es un campo de salida en la llamada MQGET.

El valor inicial de este campo es OLUNDF. Este campo se ignora si MDVER es menor que MDVER2.

MDPAN (serie de caracteres de 28 bytes)

Nombre de la aplicación que transfiere el mensaje.

Forma parte del *contexto de origen* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

El formato del MDPAN depende del valor de MDPAT.

Cuando el gestor de colas establece este campo (es decir, para todas las opciones excepto PMSETA), se establece en un valor determinado por el entorno:

- z/OS En z/OS, el gestor de colas utiliza:
  - Para el proceso por lotes z/OS , el nombre de trabajo de 8 caracteres de la tarjeta JES JOB
  - Para TSO, el identificador de usuario TSO de 7 caracteres
  - Para CICS, el ID de aplicación de 8 caracteres, seguido del ID de aplicación de 4 caracteres
  - Para IMS, el identificador del sistema IMS de 8 caracteres, seguido del nombre PSB de 8 caracteres
  - Para XCF, el nombre de grupo XCF de 8 caracteres, seguido del nombre de miembro XCF de 16 caracteres
  - Para un mensaje generado por un gestor de colas, los primeros 28 caracteres del nombre del gestor de colas
  - Para colas distribuidas sin CICS, el nombre de trabajo de 8 caracteres del iniciador de canal seguido del nombre de 8 caracteres del módulo que se coloca en la cola de mensajes no entregados seguido de un identificador de tarea de 8 caracteres.

– Para el proceso de enlaces de lenguaje MQSeries Java con IBM MQ for z/OS el nombre de trabajo de 8 caracteres del espacio de direcciones creado para el entorno de UNIX System Services. Normalmente, será un identificador de usuario TSO con un único carácter numérico añadido.

El nombre o los nombres se rellenan cada uno a la derecha con espacios en blanco, al igual que cualquier espacio en el resto del campo. Cuando hay más de un nombre, no hay ningún separador entre ellos.

- Windows En sistemas PC DOS y Windows , el gestor de colas utiliza:
  – Para una aplicación CICS , el nombre de transacción CICS
  – Para una aplicación que no es deCICS , los 28 caracteres más a la derecha del nombre completo del ejecutable
- IBM i En IBM i, el gestor de colas utiliza el nombre de trabajo completo.
- UNIX En UNIX, el gestor de colas utiliza:
  – Para una aplicación CICS , el nombre de transacción CICS
  – Para una aplicación que no es deCICS , los 14 caracteres más a la derecha del nombre completo del ejecutable si está disponible para el gestor de colas, y los espacios en blanco de lo contrario (por ejemplo, en AIX)
- En VSE/ESA, el gestor de colas utiliza el identificador de aplicación de 8 caracteres, seguido del identificador de red de 4 caracteres.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETA en el parámetro **PMO** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona LNPAN. El valor inicial de este campo es de 28 caracteres en blanco.

MDPAT (entero con signo de 10 dígitos)

Tipo de aplicación que ha transferido el mensaje.

Forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

MDPAT puede tener uno de los siguientes tipos estándar. Los tipos definidos por el usuario también se pueden utilizar, pero deben restringirse a los valores del rango ATUFST a través de ATULST.

a la(s)AIX
Aplicación AIX (el mismo valor que ATUNIX).

ATBRKR
el intermediario de ubicación.

a la(s)CICS
Transacción CICS .

ATCICB
CICS bridge.

ATVSE
Transacción CICS/VSE .

ATDOS
Aplicación IBM MQ MQI client en PC DOS.

ATDQM
Agente de gestor de colas distribuido.

ATGUAR
Aplicación Tándem Guardian (el mismo valor que ATNSK).

a la(s)IMS
IMS .

ATIMSB
Puente IMS .

ATJAVA
Java.

ATMVS
Aplicación MVS o TSO (el mismo valor que ATZOS).

NOTA
Lotus Notes Aplicación de agente.

ATNSK
Aplicación de kernel NonStop de Tándem.

AT390
Aplicación OS/390 (mismo valor que ATZOS).

AT400
IBM i .

ATQM
Gestor de colas.

a la(s)UNIX
UNIX .

ATVOS
Aplicación Stratus VOS.

ATWIN
Aplicación Windows de 16 bits.

ATWINT
Aplicación Windows de 32 bits.

ATXCF
XCF.

ATZOS
z/OS .

ATDEF
Tipo de aplicación predeterminado.

Es el tipo de aplicación predeterminado para la plataforma en la que se ejecuta la aplicación.

Nota: El valor de esta constante es específico del entorno.

ATUNK
Tipo de aplicación desconocido.

Este valor se puede utilizar para indicar que el tipo de aplicación es desconocido, aunque haya otra información de contexto.

ATUFST
Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST
Valor más alto para el tipo de aplicación definido por el usuario.

También se puede producir el siguiente valor especial:

ATNCON
No hay información de contexto en el mensaje.

Este valor lo establece el gestor de colas cuando se coloca un mensaje sin contexto (es decir, se especifica la opción de contexto PMNOC).

Cuando se recupera un mensaje, se puede probar MDPAT para este valor para decidir si el mensaje tiene contexto (se recomienda que MDPAT nunca esté establecido en ATNCON, por una aplicación que utilice PMSETA, si alguno de los otros campos de contexto no está en blanco).

ATSIB
Indica que un mensaje se ha originado en otro producto de mensajería de IBM MQ y ha llegado a través del puente SIB (Service Integration Bus).

Cuando el gestor de colas genera esta información como resultado de una colocación de aplicación, el campo se establece en un valor determinado por el entorno.

IBM i Tenga en cuenta que en IBM i, el campo se establece en AT400; el gestor de colas nunca utiliza ATCICS en IBM i.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETA en el parámetro **PMO** . Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el MDPAT que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDPAT que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDPAT cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDPAT en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo se establece en ATNCON.

Es un campo de salida para la llamada MQGET. El valor inicial de este campo es ATNCON.

MDPD (serie de caracteres de 8 bytes)

Fecha en que se transfirió el mensaje.

Forma parte del *contexto de origen* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- AAAAMMDD

donde los caracteres representan:

YYYY
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos MDPD y MDPT , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, la fecha es la fecha en la que se ha colocado el mensaje y no la fecha en la que se ha confirmado la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETA en el parámetro **PMO** . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el MDPD que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDPD que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDPD cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDPD en

todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona LNPDAT. El valor inicial de este campo es de 8 caracteres en blanco.

MDPER (entero con signo de 10 dígitos)

Persistencia de mensajes.

Esto indica si el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Para las llamadas MQPUT y MQPUT1 , el valor debe ser uno de los siguientes:

PEPER
El mensaje es persistente.

Esto significa que el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Una vez que se ha colocado el mensaje y se ha confirmado la unidad de trabajo del putter (si el mensaje se coloca como parte de una unidad de trabajo), el mensaje se conserva en el almacenamiento auxiliar. Permanece allí hasta que se elimina el mensaje de la cola y se confirma la unidad de trabajo del método getter (si el mensaje se recupera como parte de una unidad de trabajo).

Cuando se envía un mensaje persistente a una cola remota, se utiliza un mecanismo de almacenamiento y reenvío para retener el mensaje en cada gestor de colas a lo largo de la ruta al destino, hasta que se sepa que el mensaje ha llegado al siguiente gestor de colas.

Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas donde el nivel de estructura del recurso de asociación es menor que tres, o la estructura del recurso de asociación no es recuperable.

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes, colas predefinidas y colas compartidas donde el nivel de estructura del recurso de acoplamiento es 3 y el recurso de acoplamiento es recuperable.

PENDIENTE
El mensaje no es persistente.

Esto significa que el mensaje normalmente no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante el reinicio del gestor de colas.

En el caso especial de las colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a las anomalías del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

PEQDEF
El mensaje tiene persistencia predeterminada.

- Si la cola es una cola de clúster, la persistencia del mensaje se toma del atributo **DefPersistence** definido en el gestor de colas de destino que es propietario de la instancia concreta de la cola en la que se coloca el mensaje. Normalmente, todas las instancias de una cola de clúster tienen el mismo valor para el atributo **DefPersistence** , aunque no es obligatorio.

  El valor de **DefPersistence** se copia en el campo *MDPER* cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia **DefPersistence** , los mensajes que ya se han colocado en la cola no se verán afectados.
- Si la cola no es una cola de clúster, la persistencia del mensaje se toma del atributo **DefPersistence** definido en el gestor de colas local, incluso si el gestor de colas de destino es remoto.

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la primera definición de la vía de acceso. Puede ser lo siguiente:

– una cola alias
– Una cola local
– Una definición local de una cola remota
– Un alias de gestor de colas
– Una cola de transmisión (por ejemplo, la cola `DefXmitQName`)

El valor de **`DefPersistence`** se copia en el campo `MDPER` cuando se coloca el mensaje. Si posteriormente se cambia **`DefPersistence`** , los mensajes que ya se han colocado no se verán afectados.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Al responder a un mensaje, las aplicaciones normalmente deben utilizar para el mensaje de respuesta la persistencia del mensaje de solicitud.

Para una llamada MQGET, el valor devuelto es PEPER o PENPER.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es PEQDEF.

MDPRI (entero con signo de 10 dígitos)

Prioridad del mensaje.

Para las llamadas MQPUT y MQPUT1 , el valor debe ser mayor o igual que cero; cero es la prioridad más baja. También se puede utilizar el siguiente valor especial:

PRQDEF
Prioridad predeterminada para la cola.

- Si la cola es una cola de clúster, la prioridad del mensaje se toma del atributo **`DefPriority`** tal como se define en el gestor de colas de destino que es propietario de la instancia concreta de la cola en la que se coloca el mensaje. Normalmente, todas las instancias de una cola de clúster tienen el mismo valor para el atributo **`DefPriority`** , aunque no es obligatorio.

  El valor de **`DefPriority`** se copia en el campo `MDPRI` cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia **`DefPriority`** , los mensajes que ya se han colocado en la cola no se verán afectados.
- Si la cola no es una cola de clúster, la prioridad del mensaje se toma del atributo **`DefPriority`** tal como se define en el gestor de colas local, incluso si el gestor de colas de destino es remoto.

  Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada se toma del valor de este atributo en la primera definición de la vía de acceso. Puede ser lo siguiente:

  – una cola alias
  – Una cola local
  – Una definición local de una cola remota
  – Un alias de gestor de colas
  – Una cola de transmisión (por ejemplo, la cola `DefXmitQName`)

  El valor de **`DefPriority`** se copia en el campo `MDPRI` cuando se coloca el mensaje. Si posteriormente se cambia **`DefPriority`** , los mensajes que ya se han colocado no se verán afectados.

El valor devuelto por la llamada MQGET es siempre mayor o igual que cero; el valor PRQDEF nunca se devuelve.

Si un mensaje se coloca con una prioridad mayor que el máximo soportado por el gestor de colas local (este máximo lo proporciona el atributo de gestor de colas **MaxPriority**), el mensaje lo acepta el gestor de colas, pero se coloca en la cola con la prioridad máxima del gestor de colas; la llamada MQPUT o MQPUT1 se completa con CCWARN y el código de razón RC2049. Sin embargo, el campo MDPRI conserva el valor especificado por la aplicación que ha colocado el mensaje.

Al responder a un mensaje, las aplicaciones normalmente deben utilizar para el mensaje de respuesta la prioridad del mensaje de solicitud. En otras situaciones, especificar PRQDEF permite que el ajuste de prioridad se lleve a cabo sin cambiar la aplicación.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es PRQDEF.

MDPT (serie de caracteres de 8 bytes)

Hora en que se transfirió el mensaje.

Forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan (en orden):

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte nota)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, es posible que en raras ocasiones se devuelvan 60 o 61 durante los segundos en MDPT. Esto sucede cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos MDPD y MDPT , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, el tiempo es el momento en que se colocó el mensaje, y no el momento en que se comprometió la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETA en el parámetro **PMO** . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Después de la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el valor MDPT que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de MDPT que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como MDPT cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente MDPT en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona LNPTIM. El valor inicial de este campo es de 8 caracteres en blanco.

MDREP (entero con signo de 10 dígitos)

Opciones para los mensajes de informe.

Un mensaje de informe es un mensaje sobre otro mensaje, utilizado para informar a una aplicación sobre sucesos esperados o inesperados relacionados con el mensaje original. El campo MDREP permite a la aplicación que envía el mensaje original especificar qué mensajes de informe son necesarios, si los datos de mensaje de aplicación se van a incluir en ellos y también (para informes y respuestas) cómo se van a establecer los identificadores de mensaje y correlación en el informe o mensaje de respuesta. Se puede solicitar cualquiera o todos (o ninguno) de los siguientes tipos de mensajes de informe:

- Excepción
- Caducidad
- Confirmar al llegar (COA)
- Confirmar en entrega (COD)
- notificación de acción positiva (PAN)
- notificación de acción negativa (NAN)

Si se necesita más de un tipo de mensaje de informe, o se necesitan otras opciones de informe, los valores se pueden añadir juntos (no añada la misma constante más de una vez).

La aplicación que recibe el mensaje de informe puede determinar la razón por la que se ha generado el informe examinando el campo MDFB en el MQMD; consulte el campo MDFB para obtener más detalles.

El uso de opciones de informe al colocar un mensaje en un tema puede hacer que se generen cero, uno o varios mensajes de informe y se envíen a la aplicación. Esto se debe a que el mensaje de publicación se puede enviar a cero, una o muchas aplicaciones de suscripción.

Opciones de excepción: puede especificar una de las opciones siguientes para solicitar un mensaje de informe de excepción.

ROACTIVIDAD

Informes de actividad necesarios

Esta opción de informe permite generar un informe de actividad, siempre que las aplicaciones de soporte procesen un mensaje con este conjunto de opciones de informe.

Los mensajes con esta opción de informe establecida deben ser aceptados por cualquier gestor de colas, incluso si no 'entienden' la opción. Esto permite que la opción de informe se establezca en cualquier mensaje de usuario, incluso si los procesan los gestores de colas anteriores. Para conseguirlo, la opción de informe se coloca en el subcampo ROAUM.

Si un proceso (ya sea un gestor de colas o un proceso de usuario) realiza una actividad en un mensaje con ROACT establecido, puede elegir generar y colocar un informe de actividad.

La opción de informe de actividad permite rastrear la ruta de cualquier mensaje a través de una red de gestores de colas. La opción de informe se puede especificar en cualquier mensaje de usuario actual e instantáneamente pueden empezar a calcular la ruta del mensaje a través de la red. Si la aplicación que genera el mensaje no puede habilitar la generación de informes de actividad, se puede habilitar utilizando una salida cruzada de API proporcionada por los administradores del gestor de colas.

Se aplican varias condiciones a los informes de actividad:

1. La ruta será menos detallada si hay menos gestores de colas en la red que puedan generar informes de actividad.
2. Los informes de actividad pueden no ser fácilmente "ordenables" para determinar la ruta tomada.
3. Es posible que los informes de actividad no puedan encontrar una ruta a su destino solicitado.

ROEXC

Se necesitan informes de excepción.

Un agente de canal de mensajes puede generar este tipo de informe cuando se envía un mensaje a otro gestor de colas y el mensaje no se puede entregar a la cola de destino especificada. Por ejemplo, la cola de destino o una cola de transmisión intermedia puede estar llena, o el mensaje puede ser demasiado grande para la cola.

La generación del mensaje de informe de excepción depende de la persistencia del mensaje original y de la velocidad del canal de mensajes (normal o rápido) a través del cual viaja el mensaje original:

- Para todos los mensajes persistentes y para los mensajes no persistentes que viajan a través de canales de mensajes normales, el informe de excepción sólo se genera si la acción especificada por la aplicación emisora para la condición de error se puede completar correctamente. La aplicación emisora puede especificar una de las acciones siguientes para controlar la disposición del mensaje original cuando se produce la condición de error:
  - RODLQ (esto hace que el mensaje original se coloque en la cola de mensajes no entregados).
  - RODISC (esto hace que se descarte el mensaje original).

  Si la acción especificada por la aplicación emisora no se puede completar correctamente, el mensaje original se deja en la cola de transmisión y no se genera ningún mensaje de informe de excepción.
- Para los mensajes no persistentes que viajan a través de canales de mensajes rápidos, el mensaje original se elimina de la cola de transmisión y el informe de excepción se genera incluso si la acción especificada para la condición de error no se puede completar correctamente. Por ejemplo, si se especifica RODLQ, pero el mensaje original no se puede colocar en la cola de mensajes no entregados porque (digamos) esa cola está llena, se genera el mensaje de informe de excepción y se descarta el mensaje original.

  Consulte Persistencia de mensajes para obtener más información sobre los canales de mensajes normales y rápidos.

No se genera un informe de excepción si la aplicación que ha colocado el mensaje original puede recibir una notificación síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 .

Las aplicaciones también pueden enviar informes de excepción, para indicar que un mensaje que ha recibido no se puede procesar (por ejemplo, porque es una transacción de débito que haría que la cuenta excediera su límite de crédito).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

ROEXCD

Informes de excepción con datos necesarios.

Esto es lo mismo que ROEXC, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

ROEXCF

Informes de excepción con datos completos necesarios.

Esto es lo mismo que ROEXC, excepto que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

Opciones de caducidad: puede especificar una de las opciones siguientes para solicitar un mensaje de informe de caducidad.

ROEXP
Informes de caducidad necesarios.

Este tipo de informe lo genera el gestor de colas si el mensaje se descarta antes de la entrega a una aplicación porque ha pasado su hora de caducidad (consulte el campo MDEXP). Si esta opción no está establecida, no se genera ningún mensaje de informe si se descarta un mensaje por esta razón (incluso si se especifica una de las opciones ROEXC*).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

ROEXPD
Informes de caducidad con datos necesarios.

Es lo mismo que ROEXP, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

ROEXPF
Informes de caducidad con datos completos necesarios.

Esto es lo mismo que ROEXP, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

Opciones de confirmación al llegar: puede especificar una de las opciones siguientes para solicitar un mensaje de informe de confirmación al llegar.

ROCOA
Informes de confirmación de llegada necesarios.

Este tipo de informe lo genera el gestor de colas propietario de la cola de destino, cuando el mensaje se coloca en la cola de destino. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se coloca como parte de una unidad de trabajo, y la cola de destino es una cola local, el mensaje de informe COA generado por el gestor de colas pasa a estar disponible para su recuperación sólo si y cuando se confirma la unidad de trabajo.

No se genera un informe de COA si el campo MDFMT del descriptor de mensaje es FMXQH o FMDLH. Esto impide que se genere un informe de COA si el mensaje se coloca en una cola de transmisión o no se puede entregar y se coloca en una cola de mensajes no entregados.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

ROCOAD
Confirme los informes de llegada con los datos necesarios.

Es lo mismo que ROCOA, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

ROCOAF
Informes de confirmación de llegada con datos completos necesarios.

Es lo mismo que ROCOA, excepto que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

Opciones de descarte y caducidad: puede especificar la siguiente opción para establecer la hora de caducidad y el distintivo de descarte para los mensajes de informe.

ROPDAE

Establezca la hora de caducidad del mensaje de informe y el distintivo de descarte.

Esta opción garantiza que los mensajes de informe y los mensajes de respuesta hereden la hora de caducidad y el distintivo de descarte (descarte o no) de sus mensajes originales. Con esta opción establecida, los mensajes de informe y respuesta:

1. Hereda el distintivo RODISC (si se ha establecido).
2. Hereda el tiempo de caducidad restante del mensaje, si el mensaje no es un informe de caducidad. Si el mensaje es un informe de caducidad, el tiempo de caducidad se establece en 60 segundos.

Con esta opción establecida, se aplica lo siguiente:

Nota:

1. Los mensajes de informe y respuesta se generan con un distintivo de descarte y un valor de caducidad, y no pueden permanecer en el sistema.
2. Se impide que los mensajes de ruta de rastreo lleguen a las colas de destino en gestores de colas habilitados para ruta no de rastreo.
3. Se impide que las colas se llenen con informes que no se pueden entregar, si se rompen los enlaces de comunicaciones.
4. Las respuestas del servidor de mandatos heredan la caducidad restante de la solicitud.

Opciones de confirmación al entregar: puede especificar una de las opciones siguientes para solicitar un mensaje de informe de confirmación al entregar.

ROCOD

Se necesitan informes de confirmación al entregar.

Este tipo de informe lo genera el gestor de colas cuando una aplicación recupera el mensaje de la cola de destino de una forma que hace que el mensaje se suprima de la cola. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se recupera como parte de una unidad de trabajo, el mensaje de informe se genera dentro de la misma unidad de trabajo, de modo que el informe no está disponible hasta que se confirma la unidad de trabajo. Si la unidad de trabajo se restituye, el informe no se envía.

No se genera un informe COD si el campo `MDFMT` del descriptor de mensaje es FMDLH. Esto impide que se genere un informe COD si el mensaje no se puede entregar y se coloca en una cola de mensajes no entregados.

ROCOD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

ROCODD

Confirme los informes de entrega con los datos necesarios.

Es lo mismo que ROCOD, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

Si se especifica GMATM en la llamada MQGET para el mensaje original y el mensaje recuperado se trunca, la cantidad de datos de mensaje de aplicación colocados en el mensaje de informe es la mínima de:

- La longitud del mensaje original
- 100 bytes.

ROCODD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

ROCODF
Informes de confirmación de entrega con datos completos necesarios.

Es lo mismo que ROCOD, excepto que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

ROCODF no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

Opciones de notificación de acción: puede especificar una o las dos opciones siguientes para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.

ROPAN
Se necesitan informes de notificación de acción positiva.

Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. Es responsabilidad de la aplicación de recuperación generar el informe si procede.

RONAN
Se necesitan informes de notificación de acción negativa.

Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje no se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe. Por ejemplo, puede ser deseable incluir algunos datos que indiquen por qué no se ha podido realizar la solicitud.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. Es responsabilidad de la aplicación de recuperación generar el informe si procede.

La determinación de qué condiciones corresponden a una acción positiva y cuáles a una acción negativa es responsabilidad de la solicitud. Sin embargo, se recomienda que si la solicitud sólo se ha realizado parcialmente, se genere un informe NAN en lugar de un informe PAN si se solicita. También se recomienda que cada condición posible debe corresponder a una acción positiva, o a una acción negativa, pero no a ambas.

Opciones de identificador de mensaje: puede especificar una de las opciones siguientes para controlar cómo se va a establecer el `MDMID` del mensaje de informe (o del mensaje de respuesta).

RONMI
Nuevo identificador de mensaje.

Esta es la acción predeterminada e indica que si se genera un informe o respuesta como resultado de este mensaje, se generará un nuevo `MDMID` para el informe o mensaje de respuesta.

ROPMI
Pasar identificador de mensaje.

Si se genera un informe o una respuesta como resultado de este mensaje, el `MDMID` de este mensaje se copiará en el `MDMID` del informe o mensaje de respuesta.

El `MsgId` de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el `MsgId` copiado en el informe o mensaje de respuesta será diferente para cada uno.

Si no se especifica esta opción, se presupone RONMI.

Opciones de identificador de correlación: puede especificar una de las opciones siguientes para controlar cómo se va a establecer el `MDCID` del mensaje de informe (o del mensaje de respuesta).

RCMTC
Copie el identificador de mensaje en el identificador de correlación.

Esta es la acción predeterminada e indica que si se genera un informe o una respuesta como resultado de este mensaje, el `MDMID` de este mensaje se copiará en el `MDCID` del informe o mensaje de respuesta.

El `MsgId` de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el `MsgId` copiado en el `CorrelId` del informe o mensaje de respuesta será diferente para cada uno.

ROPCI
Pase el identificador de correlación.

Si se genera un informe o una respuesta como resultado de este mensaje, el `MDCID` de este mensaje se copiará en el `MDCID` del informe o mensaje de respuesta.

El `MDCID` de un mensaje de publicación será específico de un suscriptor a menos que utilice la opción SOSCID y establezca el campo `SCDIC` de MQSD en CINONE. Por lo tanto, es posible que el `MDCID` copiado en el `MDCID` del informe o mensaje de respuesta sea diferente para cada uno.

Si no se especifica esta opción, se presupone ROCMTC.

Se recomienda a los servidores que respondan a las solicitudes o generen mensajes de informe que comprueben si las opciones ROPMI o ROPCI se han establecido en el mensaje original. Si lo fueran, los servidores deberían realizar la acción descrita para esas opciones. Si no se establece ninguno, los servidores deben realizar la acción predeterminada correspondiente.

: Puede especificar una de las opciones siguientes para controlar la disposición del mensaje original cuando no se puede entregar a la cola de destino. Estas opciones sólo se aplican a aquellas situaciones que generarían un mensaje de informe de excepción si la aplicación emisora hubiera solicitado uno. La aplicación puede establecer las opciones de disposición independientemente de solicitar informes de excepción.

RODLQ
Coloque el mensaje en la cola de mensajes no entregados.

Esta es la acción predeterminada e indica que el mensaje se debe colocar en la cola de mensajes no entregados, si el mensaje no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

Se generará un mensaje de informe de excepción, si el remitente lo ha solicitado.

DISCO
Descartar mensaje.

Esto indica que el mensaje se debe descartar si no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

Se generará un mensaje de informe de excepción, si el remitente lo ha solicitado.

Si es necesario devolver el mensaje original al remitente, sin colocar el mensaje original en la cola de mensajes no entregados, el remitente debe especificar RODISC con ROEXCF.

Opción predeterminada: puede especificar lo siguiente si no se necesitan opciones de informe:

RONONA
No se necesita ningún informe.

Este valor puede utilizarse para indicar que no se ha especificado ninguna otra opción. RONONE está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Información general:

1. La aplicación que envía el mensaje original debe solicitar específicamente todos los tipos de informe necesarios. Por ejemplo, si se solicita un informe de COA pero no se solicita un informe de excepción, se genera un informe de COA cuando el mensaje se coloca en la cola de destino, pero no se genera ningún informe de excepción si la cola de destino está llena cuando el mensaje llega allí. Si no se establece ninguna opción `MDREP` , el gestor de colas o el agente de canal de mensajes (MCA) no genera ningún mensaje de informe.

   Se pueden especificar algunas opciones de informe aunque el gestor de colas local no las reconozca; esto es útil cuando el gestor de colas de destino debe procesar la opción. Consulte "Opciones de informe y distintivos de mensaje en IBM i" en la página 1469 para obtener más detalles.

   Si se solicita un mensaje de informe, el nombre de la cola a la que se debe enviar el informe debe especificarse en el campo `MDRQ` . Cuando se recibe un mensaje de informe, la naturaleza del informe se puede determinar examinando el campo `MDFB` en el descriptor de mensaje.
2. Si el gestor de colas o MCA que genera un mensaje de informe no puede colocar el mensaje de informe en la cola de respuestas (por ejemplo, porque la cola de respuestas o la cola de transmisión está llena), el mensaje de informe se coloca en su lugar en la cola de mensajes no entregados. Si esto también falla, o no hay ninguna cola de mensajes no entregados, la acción realizada depende del tipo de mensaje de informe:

   - Si el mensaje de informe es un informe de excepción, el mensaje que ha hecho que se genere el informe de excepción se deja en su cola de transmisión; esto garantiza que no se pierda el mensaje.
   - Para todos los demás tipos de informe, el mensaje de informe se descarta y el proceso continúa normalmente. Esto se hace porque el mensaje original ya se ha entregado de forma segura (para los mensajes de informe COA o COD), o ya no es de ningún interés (para un mensaje de informe de caducidad).

   Una vez que un mensaje de informe se ha colocado correctamente en una cola (ya sea la cola de destino o una cola de transmisión intermedia), el mensaje ya no está sujeto a un proceso especial; se trata igual que cualquier otro mensaje.
3. Cuando se genera el informe, se abre la cola `MDRQ` y el mensaje de informe se coloca utilizando la autorización del `MDUID` en el MQMD del mensaje que provoca el informe, excepto en los casos siguientes:

   - Los informes de excepción generados por un MCA receptor se colocan con la autorización que el MCA utilizó cuando intentó colocar el mensaje que causaba el informe. El atributo de canal `CDPA` determina el identificador de usuario utilizado.
   - Los informes de COA generados por el gestor de colas se colocan con la autorización que se haya utilizado cuando el mensaje que ha provocado el informe se ha colocado en el gestor de colas que genera el informe. Por ejemplo, si el mensaje ha sido transferido por un MCA receptor utilizando el identificador de usuario del MCA, el gestor de colas coloca el informe COA utilizando el identificador de usuario del MCA.

   Las aplicaciones que generan informes normalmente deben utilizar la misma autorización que habrían utilizado para generar una respuesta; normalmente, esta debe ser la autorización del identificador de usuario en el mensaje original.

   Si el informe tiene que viajar a un destino remoto, los remitentes y receptores pueden decidir si lo aceptan, del mismo modo que lo hacen para otros mensajes.

4. Si se solicita un mensaje de informe con datos:
   - El mensaje de informe siempre se genera con la cantidad de datos solicitados por el remitente del mensaje original. Si el mensaje de informe es demasiado grande para la cola de respuestas, se produce el proceso descrito anteriormente; el mensaje de informe nunca se trunca para que quepa en la cola de respuestas.
   - Si el MDFMT del mensaje original es FMXQH, los datos incluidos en el informe no incluyen MQXQH. Los datos del informe empiezan con el primer byte de los datos más allá de MQXQH en el mensaje original. Esto ocurre si la cola es una cola de transmisión.
5. Si se recibe un mensaje de informe de COA, COD o caducidad en la cola de respuestas, se garantiza que el mensaje original ha llegado, se ha entregado o ha caducado, según corresponda. Sin embargo, si se solicita uno o más de estos mensajes de informe y no se recibe, no se puede suponer lo contrario, ya que puede haber ocurrido uno de los siguientes:
   a. El mensaje de informe se retiene porque un enlace está inactivo.
   b. El mensaje de informe se retiene porque existe una condición de bloqueo en una cola de transmisión intermedia o en la cola de respuestas (por ejemplo, la cola está llena o inhibida para colocaciones).
   c. El mensaje de informe está en una cola de mensajes no entregados.
   d. Cuando el gestor de colas intentaba generar el mensaje de informe, no pudo colocarlo en la cola adecuada y tampoco pudo colocarlo en la cola de mensajes no entregados, por lo que no se pudo generar el mensaje de informe.
   e. Se ha producido una anomalía del gestor de colas entre la acción que se notifica (llegada, entrega o caducidad) y la generación del mensaje de informe correspondiente. (Esto no sucede para los mensajes de informe COD si la aplicación recupera el mensaje original dentro de una unidad de trabajo, ya que el mensaje de informe COD se genera dentro de la misma unidad de trabajo.)

   Los mensajes de informe de excepción se pueden retener de la misma forma por las razones 1, 2 y 3 anteriores. Sin embargo, cuando un MCA no puede generar un mensaje de informe de excepción (el mensaje de informe no se puede colocar en la cola de respuestas ni en la cola de mensajes no entregados), el mensaje original permanece en la cola de transmisión del emisor y el canal se cierra. Esto se produce independientemente de si el mensaje de informe se iba a generar en el extremo emisor o receptor del canal.
6. Si el mensaje original se bloquea temporalmente (lo que genera un mensaje de informe de excepción y el mensaje original se coloca en una cola de mensajes no entregados), pero el bloqueo se borra y una aplicación lee el mensaje original de la cola de mensajes no entregados y lo coloca de nuevo en su destino, puede ocurrir lo siguiente:
   - Aunque se ha generado un mensaje de informe de excepción, el mensaje original finalmente llega correctamente a su destino.
   - Se genera más de un mensaje de informe de excepción con respecto a un único mensaje original, ya que el mensaje original puede encontrar otro bloqueo más adelante.

Informar de mensajes al transferir a un tema:

1. Se pueden generar informes al colocar un mensaje en un tema. Este mensaje se enviará a todos los suscriptores del tema, que podría ser cero, uno o varios. Esto debe tenerse en cuenta al elegir utilizar las opciones de informe, ya que se podrían generar muchos mensajes de informe como resultado.
2. Al colocar un mensaje en un tema, puede haber muchas colas de destino a las que se debe dar una copia del mensaje. Si algunas de estas colas de destino tienen un problema, como la cola llena, la finalización satisfactoria de MQPUT depende del valor de NPMSGDLV o PMSGDLV (en función de la persistencia del mensaje). Si el valor es tal que la entrega de mensajes a la cola de destino debe ser satisfactoria (por ejemplo, es un mensaje persistente para un suscriptor duradero y PMSGDLV se establece en ALL o ALLDUR), el éxito se define como uno de los siguientes criterios que se cumplen:

- Se ha colocado correctamente en la cola de suscriptores
- Uso de RODLQ y una colocación satisfactoria en la cola de mensajes no entregados si la cola del suscriptor no puede tomar el mensaje
- Uso de RODISC si la cola de suscriptores no puede tomar el mensaje.

Mensajes de informe para segmentos de mensajes:

1. Se pueden solicitar mensajes de informe para los mensajes que tienen la segmentación permitida (consulte la descripción del distintivo MFSEGA). Si el gestor de colas considera necesario segmentar el mensaje, se puede generar un mensaje de informe para cada uno de los segmentos que posteriormente encuentren la condición relevante. Por lo tanto, las aplicaciones deben estar preparadas para recibir varios mensajes de informe para cada tipo de mensaje de informe solicitado. El campo `MDGID` del mensaje de informe se puede utilizar para correlacionar los varios informes con el identificador de grupo del mensaje original y el campo `MDFB` utilizado para identificar el tipo de cada mensaje de informe.
2. Si se utiliza GMLOGO para recuperar mensajes de informe para segmentos, tenga en cuenta que las llamadas MQGET sucesivas pueden devolver informes de distintos tipos. Por ejemplo, si se solicitan los informes COA y COD para un mensaje segmentado por el gestor de colas, las llamadas MQGET para los mensajes de informe pueden devolver los mensajes de informe COA y COD intercalados de forma imprevisible. Esto se puede evitar utilizando la opción GMCMPM (opcionalmente con GMATM). GMCMPM hace que el gestor de colas vuelva a ensamblar los mensajes de informe que tienen el mismo tipo de informe. Por ejemplo, la primera llamada MQGET podría volver a ensamblar todos los mensajes COA relacionados con el mensaje original, y la segunda llamada MQGET podría volver a ensamblar todos los mensajes COD. Lo que se reensambla primero depende del tipo de mensaje de informe que se produzca primero en la cola.
3. Las aplicaciones que ellas mismas colocan segmentos pueden especificar diferentes opciones de informe para cada segmento. Sin embargo, hay que señalar los siguientes puntos:
   - Si los segmentos se recuperan utilizando la opción GMCMPM, el gestor de colas solo respeta las opciones de informe del primer segmento.
   - Si los segmentos se recuperan uno por uno, y la mayoría de ellos tienen una de las opciones ROCOD*, pero al menos un segmento no lo hace, no será posible utilizar la opción GMCMPM para recuperar los mensajes de informe con una sola llamada MQGET, o utilizar la opción GMASGA para detectar cuándo han llegado todos los mensajes de informe.
4. En una red MQ , es posible que los gestores de colas tengan prestaciones diferentes. Si un mensaje de informe para un segmento es generado por un gestor de colas o MCA que no soporta la segmentación, el gestor de colas o MCA no incluirá de forma predeterminada la información de segmento necesaria en el mensaje de informe, y esto puede dificultar la identificación del mensaje original que ha hecho que se genere el informe. Esta dificultad se puede evitar solicitando datos con el mensaje de informe, es decir, especificando las opciones RO* D o RO* F adecuadas. Sin embargo, tenga en cuenta que si se especifica RO* D, se pueden devolver menos de 100 bytes de datos de mensaje de aplicación a la aplicación que recupera el mensaje de informe, si el mensaje de informe lo genera un gestor de colas o MCA que no soporta la segmentación.

Contenido del descriptor de mensaje para un mensaje de informe: cuando el gestor de colas o el agente de canal de mensajes (MCA) genera un mensaje de informe, establece los campos del descriptor de mensaje en los valores siguientes y, a continuación, coloca el mensaje de la forma normal.

Tabla 708. Valores utilizados para los campos MQMD cuando se genera un mensaje de informe generado por el sistema

Campo de MQMD	Valor utilizado
MDSID	MDSIDV
MDVER	MDVER2
MDREP	RONONA

Tabla 708. Valores utilizados para los campos MQMD cuando se genera un mensaje de informe generado por el sistema (continuación)

Campo de MQMD	Valor utilizado
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	Según proceda por la naturaleza del informe (FBCOA, FBCOD, FBEXP o un valor RC*)
MDENC	Copiado del descriptor de mensaje original
MDCSI	Copiado del descriptor de mensaje original
MDFMT	Copiado del descriptor de mensaje original
MDPRI	Copiado del descriptor de mensaje original
MDPER	Copiado del descriptor de mensaje original
MDMID	Según lo especificado por las opciones de informe en el descriptor de mensaje original
MDCID	Según lo especificado por las opciones de informe en el descriptor de mensaje original
MDBOC	0
MDRQ	Espacios en blanco
MDRM	Nombre de gestor de colas
MDUID	Tal como lo establece la opción PMPASI
MDACC	Tal como lo establece la opción PMPASI
MDAID	Tal como lo establece la opción PMPASI
MDPAT	ATQM, o según corresponda para el agente de canal de mensajes
MDPAN	Primeros 28 bytes del nombre del gestor de colas o del nombre del agente de canal de mensajes. Para los mensajes de informe generados por el puente IMS , este campo contiene el nombre de grupo XCF y el nombre de miembro XCF del sistema IMS con el que se relaciona el mensaje.
MDPD	Fecha en la que se envía el mensaje de informe
MDPT	Hora a la que se envía el mensaje de informe
MDAOD	Espacios en blanco
MDGID	Copiado del descriptor de mensaje original
MDSEQ	Copiado del descriptor de mensaje original
MDOFF	Copiado del descriptor de mensaje original
MDMFL	Copiado del descriptor de mensaje original
MDOLN	Copiado del descriptor de mensaje original si no es OLUNDF, y establecido en la longitud de los datos del mensaje original de lo contrario

Se recomienda una aplicación que genere un informe para establecer valores similares, excepto para lo siguiente:

- El campo MDRM se puede establecer en blancos (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando se coloque el mensaje).

- Los campos de contexto deben establecerse utilizando la opción que se habría utilizado para una respuesta, normalmente PMPASI.

Análisis del campo de informe: el campo MDREP contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas en "Análisis del campo de informe en IBM i" en la página 1471.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es RONONE.

MDRM (serie de caracteres de 48 bytes)

Nombre del gestor de colas de respuesta.

Es el nombre del gestor de colas al que se debe enviar el mensaje de respuesta o el mensaje de informe. MDRQ es el nombre local de una cola definida en este gestor de colas.

Si el campo MDRM está en blanco, el gestor de colas local busca el nombre **MDRQ** en sus definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor **MDRM** del mensaje transmitido se sustituye por el valor del atributo **RemoteQMgrName** de la definición de la cola remota, y este valor se devolverá en el descriptor de mensaje cuando la aplicación receptora emita una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, el MDRM que se transmite con el mensaje es el nombre del gestor de colas local.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, sin embargo, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para los gestores de colas, o que este nombre es conocido por el gestor de colas emisor; esto también es cierto para el nombre transmitido, si el **MDRM** se sustituye en el mensaje transmitido.

Si no es necesaria una cola de respuesta, se recomienda (aunque no está seleccionada) que el campo MDRM se establezca en blancos; el campo no se debe dejar sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona LNQMN. El valor inicial de este campo es de 48 caracteres en blanco.

MDRQ (serie de caracteres de 48 bytes)

Nombre de la cola de respuesta.

Es el nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud de obtención del mensaje debe enviar los mensajes MTRPLY y MTRPRT. El nombre es el nombre local de una cola definida en el gestor de colas identificado por MDRM. Esta cola no debe ser una cola modelo, aunque el gestor de colas emisor no lo verifica cuando se coloca el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo no debe estar en blanco si el campo MDMT tiene el valor MTRQST, o si el campo MDREP solicita algún mensaje de informe. Sin embargo, el valor especificado (o sustituido) se pasa a la aplicación que emite la solicitud get para el mensaje, sea cual sea el tipo de mensaje.

Si el campo MDRM está en blanco, el gestor de colas local busca el nombre MDRQ en sus propias definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor MDRQ del mensaje transmitido se sustituye por el valor del atributo **RemoteQName** de la definición de la cola remota, y este valor se devolverá en el descriptor de mensaje cuando la aplicación receptora emita una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, MDRQ no se modifica.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. Sin embargo, de lo contrario, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para las colas; esto también es cierto para el nombre transmitido, si el MDRQ se sustituye en el mensaje transmitido.

La única comprobación realizada es que se ha especificado un nombre, si las circunstancias lo requieren.

Si no es necesaria una cola de respuesta, se recomienda (aunque no está seleccionada) que el campo MDRQ se establezca en blancos; el campo no se debe dejar sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Si un mensaje que requiere un mensaje de informe no se puede entregar, y el mensaje de informe tampoco se puede entregar a la cola especificada, tanto el mensaje original como el mensaje de informe van a la cola de mensajes no entregados. Consulte el atributo **DeadLetterQName** descrito en "Atributos del gestor de colas en IBM i" en la página 1434.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

MDSEQ (entero con signo de 10 dígitos)

Número de secuencia del mensaje lógico en el grupo.

Los números de secuencia empiezan en 1 y aumentan en 1 para cada nuevo mensaje lógico del grupo, hasta un máximo de 999 999 999. Un mensaje físico que no está en un grupo tiene un número de secuencia de 1.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se ha especificado MOSEQN.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que MDSEQ esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

En la entrada a la llamada MQGET, el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es uno. Este campo se ignora si MDVER es menor que MDVER2.

MDSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

MDSIDV
Identificador de la estructura del descriptor de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MDSIDV.

MDUID (serie de caracteres de 12 bytes)

Identificador de usuario.

Forma parte del *contexto de identidad* del mensaje. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

MDUID especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato.

Después de recibir un mensaje, se puede utilizar MDUID en el campo ODAU del parámetro **OBJDSC** de una llamada MQOPEN o MQPUT1 posterior, para que se realice la comprobación de autorización para el usuario MDUID en lugar de la aplicación que realiza la apertura.

Cuando el gestor de colas genera esta información para una llamada MQPUT o MQPUT1 , el gestor de colas utiliza un identificador de usuario determinado desde el entorno.

Cuando el identificador de usuario se determina desde el entorno:

- z/OS En z/OS, el gestor de colas utiliza:
  - Para el proceso por lotes, el identificador de usuario de la tarjeta JES JOB o tarea iniciada
  - Para TSO, el identificador de usuario de inicio de sesión
  - Para CICS, el identificador de usuario asociado a la tarea
  - Para IMS, el identificador de usuario depende del tipo de aplicación:
    - Durante:
      - Regiones BMP no de mensaje
      - Regiones IFP no de mensajes
      - Mensaje BMP y regiones IFP de mensaje que no han emitido una llamada de GU satisfactoria

      el gestor de colas utiliza el identificador de usuario de la tarjeta JES JOB de la región o el identificador de usuario TSO. Si están en blanco o son nulos, utiliza el nombre del bloque de especificación de programa (PSB).
    - Durante:
      - Mensaje BMP y regiones IFP de mensaje que han emitido una llamada de GU satisfactoria
      - Regiones MPP

      el gestor de colas utiliza uno de los siguientes:
      - El identificador de usuario conectado asociado con el mensaje
      - El nombre del terminal lógico (LTERM)
      - El identificador de usuario de la tarjeta JES JOB de la región
      - El identificador de usuario TSO
      - El nombre de PSB
- IBM i En IBM i, el gestor de colas utiliza el nombre del perfil de usuario asociado con el trabajo de aplicación.
- UNIX En UNIX, el gestor de colas utiliza:
  - El nombre de inicio de sesión de la aplicación
  - Identificador de usuario efectivo del proceso si no hay ningún inicio de sesión disponible
  - El identificador de usuario asociado a la transacción, si la aplicación es una transacción CICS
- En VSE/ESA, es un campo reservado.
- Windows En Windows, el gestor de colas utiliza los primeros 12 caracteres del nombre de usuario conectado.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica PMSETI o PMSETA en el parámetro **PMO** . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica PMSETI o PMSETA, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el `MDUID` que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de `MDUID` que se conserva con el mensaje si se conserva (consulte la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como `MDUID` cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar

temporalmente MDUID en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona LNUID. El valor inicial de este campo es de 12 caracteres en blanco.

MDVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

MDVER1
Estructura del descriptor de mensaje Version-1 .

MDVER2
Estructura del descriptor de mensaje Version-2 .

Nota: Cuando se utiliza un MQMD version-2 , el gestor de colas realiza comprobaciones adicionales en cualquier estructura de cabecera MQ que pueda estar presente al principio de los datos del mensaje de aplicación; para obtener más detalles, consulte las notas de uso de la llamada MQPUT.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MDVERC
Versión actual de la estructura del descriptor de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MDVER1.

Valores iniciales

Tabla 709. Valores iniciales de campos en MQMD

Nombre de campo	Nombre de constante	Valor de constante
MDSID	MDSIDV	'MD¬¬'
MDVER	MDVER1	1
MDREP	RONONA	0
MDMT	MTDGRM	8
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT	Depende del entorno
MDCSI	CSQM	0
MDFMT	FMNONE	Espacios en blanco
MDPRI	PRQDEF	-1
MDPER	PEQDEF	2
MDMID	MINONA	Nulos
MDCID	CINONA	Nulos
MDBOC	Ninguna	0
MDRQ	Ninguna	Espacios en blanco
MDRM	Ninguna	Espacios en blanco

Tabla 709. Valores iniciales de campos en MQMD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
MDUID	Ninguna	Espacios en blanco
MDACC	ACNONE	Nulos
MDAID	Ninguna	Espacios en blanco
MDPAT	ATNCON	0
MDPAN	Ninguna	Espacios en blanco
MDPD	Ninguna	Espacios en blanco
MDPT	Ninguna	Espacios en blanco
MDAOD	Ninguna	Espacios en blanco
MDGID	GINONA	Nulos
MDSEQ	Ninguna	1
MDOFF	Ninguna	0
MDMFL	MFNONE	0
MDOLN	OLUNDF	-1
Notas: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración 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                                     00000000000000000000-
D                                     000000000000')
D* Correlation identifier
D  MDCID                 73     96    INZ(X'00000000000000-
D                                     00000000000000000000-
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 (extensión de descriptor de mensaje) en IBM i

Visión general

Finalidad: la estructura MQMDE describe los datos que a veces se producen antes de los datos del mensaje de aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2 , pero no en el MQMD version-1 .

Nombre de formato: FMMDE.

Conjunto de caracteres y codificación: Los datos de MQMDE deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación de MQMDE deben establecerse en los campos *MDCSI* y *MDENC* en:

- El MQMD (si la estructura MQMDE está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQMDE (todos los demás casos).

Si el MQMDE no está en el juego de caracteres y la codificación del gestor de colas, el MQMDE se acepta pero no se respeta, es decir, el MQMDE se trata como datos de mensaje.

Uso: las aplicaciones normales deben utilizar un MQMD version-2 , en cuyo caso no encontrarán una estructura MQMDE. Sin embargo, las aplicaciones especializadas y las aplicaciones que siguen utilizando un MQMD version-1 , pueden encontrar un MQMDE en algunas situaciones. La estructura MQMDE puede producirse en las circunstancias siguientes:

- Especificado en las llamadas MQPUT y MQPUT1
- Devuelto por la llamada MQGET
- En mensajes en colas de transmisión
- “MQMDE especificado en llamadas MQPUT y MQPUT1” en la página 1180
- “MQMDE devuelto por la llamada MQGET” en la página 1180

- “MQMDE en mensajes en colas de transmisión” en la página 1181
- “Campos” en la página 1181
- “Valores iniciales” en la página 1183
- “Declaración RPG” en la página 1183

MQMDE especificado en llamadas MQPUT y MQPUT1

En las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD version-1 , la aplicación puede opcionalmente añadir un prefijo a los datos de mensaje con un MQMDE, estableciendo el campo *MDFMT* de MQMD en FMMDE para indicar que existe un MQMDE. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE. Los valores predeterminados que utiliza el gestor de colas son los mismos que los valores iniciales para la estructura; consulte Tabla 711 en la página 1183.

Si la aplicación proporciona un version-2 MQMD *y* prefija los datos de mensaje de aplicación con un MQMDE, las estructuras se procesan tal como se muestra en la Tabla 710 en la página 1180.

Tabla 710. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o MQPUT1

MQMD Version	Valores de los campos version-2	Valores de los campos correspondientes en MQMDE	Acción realizada por el gestor de colas
1	-	Válido	MQMDE se respeta
2	Valor predeterminado	Válido	MQMDE se respeta
2	No predeterminado	Válido	MQMDE se trata como datos de mensaje
1 o 2	Cualquiera	No válido	La llamada falla con un código de razón adecuado
1 o 2	Cualquiera	MQMDE está en el juego de caracteres o codificación incorrectos, o es una versión no soportada	MQMDE se trata como datos de mensaje

Hay un caso especial. Si la aplicación utiliza un MQMD version-2 para colocar un mensaje que es un segmento (es decir, se establece el distintivo MFSEG o MFLSEG) y el nombre de formato del MQMD es FMDLH, el gestor de colas genera una estructura MQMDE y la inserta *entre* la estructura MQDLH y los datos que le siguen. En el MQMD que el gestor de colas conserva con el mensaje, los campos version-2 se establecen en sus valores predeterminados.

Varios de los campos que existen en MQMD version-2 pero no en MQMD version-1 son campos de entrada/salida en MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .

MQMDE devuelto por la llamada MQGET

En la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. El gestor de colas establece el campo *MDFMT* en MQMD en el valor FMMDE para indicar que hay un MQMDE presente.

Si la aplicación proporciona un MQMDE al inicio del parámetro **BUFFER** , el MQMDE se ignora. Al volver de la llamada MQGET, se sustituye por el MQMDE para el mensaje (si es necesario), o se sobrescribe por los datos del mensaje de aplicación (si el MQMDE no es necesario).

Si la llamada MQGET devuelve un MQMDE, los datos de MQMDE suelen estar en el juego de caracteres y la codificación del gestor de colas. Sin embargo, MQMDE puede estar en algún otro juego de caracteres y codificación si:

- MQMDE se ha tratado como datos en la llamada MQPUT o MQPUT1 (consulte Tabla 710 en la página 1180 para conocer las circunstancias que pueden causar esto).
- El mensaje se ha recibido de un gestor de colas remoto conectado mediante una conexión TCP, y el agente de canal de mensajes receptor (MCA) no se ha configurado correctamente (consulte Seguridad de objetos de IBM MQ for IBM i para obtener más información).

MQMDE en mensajes en colas de transmisión

Los mensajes de las colas de transmisión tienen como prefijo la estructura MQXQH, que contiene un MQMD version-1 . Un MQMDE también puede estar presente, situado entre la estructura MQXQH y los datos del mensaje de aplicación, pero normalmente sólo estará presente si uno o varios de los campos de MQMDE tienen un valor no predeterminado.

También se pueden producir otras estructuras de cabecera IBM MQ entre la estructura MQXQH y los datos del mensaje de aplicación. Por ejemplo, cuando la cabecera de mensaje no entregado MQDLH está presente y el mensaje no es un segmento, el orden es:

- MQXQH (que contiene un MQMD version-1)
- MQMDE
- MQDLH
- Datos de mensaje de aplicación

Campos

La estructura MQMDE contiene los campos siguientes; los campos se describen en **orden alfabético**:

MECSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen a MQMDE.

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQMDE; no se aplica a los datos de tipo carácter de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

MEENC (entero con signo de 10 dígitos)
MEENC (entero con signo de 10 dígitos)

Especifica la codificación numérica de los datos que siguen a la estructura MQMDE; no se aplica a los datos numéricos de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Consulte el campo *MDENC* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener más información sobre las codificaciones de datos.

El valor inicial de este campo es ENNAT.

MEFLG (entero con signo de 10 dígitos)

Distintivos generales.

Se puede especificar el distintivo siguiente:

MEFNON
Sin distintivos.

El valor inicial de este campo es MEFNON.

MEFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen a MQMDE.

Especifica el nombre de formato de los datos que siguen a la estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Consulte el campo *MDFMT* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener más información sobre los nombres de formato.

El valor inicial de este campo es FMNONE.

MEGID (serie de bits de 24 bytes)

Identificador de grupo.

Consulte el campo *MDGID* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133. El valor inicial de este campo es GINONE.

MELEN (entero con signo de 10 dígitos)

Longitud de la estructura MQMDE.

Se define el valor siguiente:

MELEN2
Longitud de la estructura de extensión del descriptor de mensaje version-2 .

El valor inicial de este campo es MELEN2.

MEMFL (entero con signo de 10 dígitos)

Distintivos de mensaje.

Consulte el campo *MDMFL* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133. El valor inicial de este campo es MFNONE.

MEOFF (entero con signo de 10 dígitos)

Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico.

Consulte el campo *MDOFF* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133. El valor inicial de este campo es 0.

MEOLN (entero con signo de 10 dígitos)

Longitud del mensaje original.

Consulte el campo *MDOLN* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133. El valor inicial de este campo es OLUNDF.

MESEQ (entero con signo de 10 dígitos)

Número de secuencia del mensaje lógico en el grupo.

Consulte el campo *MDSEQ* descrito en “MQMD (Descriptor de mensaje) en IBM i” en la página 1133. El valor inicial de este campo es 1.

MESID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

MESIDV
Identificador de la estructura de extensión del descriptor de mensaje.

El valor inicial de este campo es MESIDV.

MEVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

MEVER2
Estructura de extensión del descriptor de mensaje Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

MEVERC
Versión actual de la estructura de extensión del descriptor de mensaje.

El valor inicial de este campo es MEVER2.

Valores iniciales

Tabla 711. Valores iniciales de los campos en MQMDE

Nombre de campo	Nombre de constante	Valor de constante
MESID	MESIDV	'MDE¬'
MEVER	MEVER2	2
MELEN	MELEN2	72
MEENC	ENNAT	Depende del entorno
MECSI	CUNDF	0
MEFMT	FMNONE	Espacios en blanco
MEFLG	MEFNON	0
MEGID	GINONA	Nulos
MESEQ	Ninguna	1
MEOFF	Ninguna	0
MEMFL	MFNONE	0
MEOLN	OLUNDF	-1

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQMDE Structure
```

```
     D*
     D* Structure identifier
     D  MESID                  1      4    INZ('MDE ')
     D* Structure version number
     D  MEVER                  5      8I 0 INZ(2)
     D* Length of MQMDE structure
     D  MELEN                  9     12I 0 INZ(72)
     D* Numeric encoding of data that followsMQMDE
     D  MEENC                 13     16I 0 INZ(273)
     D* Character-set identifier of data thatfollows MQMDE
     D  MECSI                 17     20I 0 INZ(0)
     D* Format name of data that followsMQMDE
     D  MEFMT                 21     28    INZ('        ')
     D* General flags
     D  MEFLG                 29     32I 0 INZ(0)
     D* Group identifier
     D  MEGID                 33     56    INZ(X'00000000000000-
     D                                     00000000000000000000-
     D                                     000000000000')
     D* Sequence number of logical messagewithin group
     D  MESEQ                 57     60I 0 INZ(1)
     D* Offset of data in physical messagefrom start of logical message
     D  MEOFF                 61     64I 0 INZ(0)
     D* Message flags
     D  MEMFL                 65     68I 0 INZ(0)
     D* Length of original message
     D  MEOLN                 69     72I 0 INZ(-1)
```

IBM i MQMHBO (Descriptor de mensaje para opciones de almacenamiento intermedio) en IBM i

Estructura que define el descriptor de mensaje para las opciones de almacenamiento intermedio

Visión general

Finalidad: la estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se generan los almacenamientos intermedios a partir de los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUF.

Juego de caracteres y codificación: Los datos de MQMHBO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1184
- “Valores iniciales” en la página 1185
- “Declaración RPG” en la página 1185

Campos

La estructura MQMHBO contiene los campos siguientes; los campos se describen en **orden alfabético**:

MBOPT (entero con signo de 10 dígitos)

Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-Campo MBOPT.

Estas opciones controlan la acción de MQMHBUF.

Debe especificar la opción siguiente:

MPRRF
Al convertir propiedades de un descriptor de mensaje en un almacenamiento intermedio, conviértalas al formato MQRFH2 .

Opcionalmente, también puede especificar la opción siguiente. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

MBDLPR
Las propiedades que se añaden al almacenamiento intermedio se suprimen del descriptor de contexto de mensaje. Si la llamada falla, no se suprimen las propiedades.

Siempre es un campo de entrada. El valor inicial de este campo es MBPRRF.

MBSID (entero con signo de 10 dígitos)

Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-campo MBSID.

Es el identificador de estructura. El valor debe ser:

MBSIDV
Identificador del descriptor de mensaje para la estructura de opciones de almacenamiento intermedio.

Siempre es un campo de entrada. El valor inicial de este campo isMBSIDV.

MBVER (entero con signo de 10 dígitos)

Es el número de versión de la estructura. El valor debe ser:

MBVER1
Número de versión del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

La constante siguiente especifica el número de versión de la versión actual:

MMBVERC
Versión actual del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

Siempre es un campo de entrada. El valor inicial de este campo es MBVER1.

Valores iniciales

Tabla 712. Valores iniciales de los campos en MQMHBO

Nombre de campo	Nombre de constante	Valor de constante
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1
MBOPT	MPRRF	

Notas:

1. El valor Serie o espacios en blanco nulos indica un carácter en blanco.

Declaración 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 (Descriptor de objeto) en IBM i

La estructura MQOD se utiliza para especificar un objeto por nombre.

Visión general

Finalidad: Los siguientes tipos de objeto son válidos:

- Cola o lista de distribución
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1 .

Versión: La versión actual de MQOD es ODVER4. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQOD soportada por el entorno, pero con el valor inicial del campo *ODVER* establecido en ODVER1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *ODVER* en el número de versión de la versión necesaria.

Para abrir una lista de distribución, *ODVER* debe ser ODVER2 o superior.

Conjunto de caracteres y codificación: Los datos de MQOD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

- "Campos" en la página 1186
- "Valores iniciales" en la página 1193
- "Declaración RPG" en la página 1194

Campos

La estructura MQOD contiene los campos siguientes; los campos se describen en **orden alfabético**:

ODASI (serie de bits de 40 bytes)

Identificador de seguridad alternativo.

Se trata de un identificador de seguridad que se pasa con *ODAU* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas. *ODASI* sólo se utiliza si:

- OOALTU se especifica en la llamada MQOPEN, o
- PMALTU se especifica en la llamada MQPUT1 ,

y el campo *ODAU* no está completamente en blanco hasta el primer carácter nulo o el final del campo.

El campo *ODASI* tiene la estructura siguiente:

- El primer byte es un entero binario que contiene la longitud de los datos significativos que siguen; el valor excluye el propio byte de longitud. Si no hay ningún identificador de seguridad, la longitud es cero.
- El segundo byte indica el tipo de identificador de seguridad que está presente; son posibles los valores siguientes:

  SITWNT
  Identificador de seguridad de Windows .

SITNÓN
Sin identificador de seguridad.

- El tercer byte y los bytes subsiguientes hasta la longitud definida por el primer byte contienen el propio identificador de seguridad.
- Los bytes restantes en el campo se establecen en cero binario.

Se puede utilizar el siguiente valor especial:

SINONA
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Este es un campo de entrada. La longitud de este campo la proporciona LNSCID. El valor inicial de este campo es SINONE. Este campo se ignora si *ODVER* es menor que ODVER3.

ODAU (serie de caracteres de 12 bytes)

Identificador de usuario alternativo.

Si se especifica OOALTU para la llamada MQOPEN, o PMALTU para la llamada MQPUT1 , este campo contiene un identificador de usuario alternativo que se debe utilizar para comprobar la autorización para la apertura, en lugar del identificador de usuario con el que se ejecuta actualmente la aplicación. Sin embargo, algunas comprobaciones se siguen realizando con el identificador de usuario actual (por ejemplo, comprobaciones de contexto).

Si no se especifican OOALTU y PMALTU y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la apertura sólo puede realizarse correctamente si no se necesita autorización de usuario para abrir este objeto con las opciones especificadas.

Si no se especifica OOALTU ni PMALTU, este campo se ignora.

Este es un campo de entrada. La longitud de este campo la proporciona LNUID. El valor inicial de este campo es de 12 caracteres en blanco.

ODDN (serie de caracteres de 48 bytes)

Nombre de cola dinámica.

Es el nombre de una cola dinámica que va a crear la llamada MQOPEN. Esto sólo es relevante cuando *ODON* especifica el nombre de una cola modelo; en todos los demás casos, se ignora *ODDN* .

Los caracteres que son válidos en el nombre son los mismos que los de *ODON*, excepto que también es válido un asterisco. Un nombre que esté en blanco (o uno en el que sólo se muestren espacios en blanco antes del primer carácter nulo) no es válido si *ODON* es el nombre de una cola modelo.

Si el último carácter no en blanco del nombre es un asterisco (`*`), el gestor de colas sustituye el asterisco por una serie de caracteres que garantiza que el nombre generado para la cola es exclusivo en el gestor de colas local. Para permitir un número suficiente de caracteres para esto, el asterisco sólo es válido en las posiciones de la 1 a la 33. No debe haber caracteres que no sean espacios en blanco o un carácter nulo detrás del asterisco.

Es válido que el asterisco aparezca en la primera posición de carácter, en cuyo caso el nombre consta únicamente de los caracteres generados por el gestor de colas.

Este es un campo de entrada. La longitud de este campo la proporciona LNQN. El valor inicial de este campo es `'AMQ.*'`, rellenado con espacios en blanco.

ODIDC (entero con signo de 10 dígitos)

Número de colas que no se han podido abrir.

Es el número de colas de la lista de distribución que no se han podido abrir correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Nota: Si está presente, este campo sólo se establece si el parámetro **CMPCOD** de la llamada MQOPEN o MQPUT1 es CCOK o CCWARN; no se establece si el parámetro **CMPCOD** es CCFAIL.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODKDC (entero con signo de 10 dígitos)

Número de colas locales abiertas satisfactoriamente.

Es el número de colas de la lista de distribución que se resuelven en colas locales y que se han abierto correctamente. El recuento no incluye las colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODMN (serie de caracteres de 48 bytes)

Nombre del gestor de colas de objetos.

Es el nombre del gestor de colas en el que se define el objeto *ODON* . Los caracteres que son válidos en el nombre son los mismos que los de *ODON* (consulte anteriormente). Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ODOT* es OTTOP, OTNLST, OTPRO u OTQM, *ODMN* debe estar en blanco o el nombre del gestor de colas local.
- Si *ODON* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ODMN* el nombre del gestor de colas en el que se crea la cola; este es el nombre del gestor de colas local. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si *ODON* es el nombre de una cola de clúster y *ODMN* está en blanco, el destino real de los mensajes enviados utilizando el descriptor de contexto de cola devuelto por la llamada MQOPEN lo elige el gestor de colas (o la salida de carga de trabajo de clúster, si hay uno instalado) de la siguiente manera:
  - Si se especifica OOBNDO, el gestor de colas selecciona una instancia de la cola de clúster durante el proceso de la llamada MQOPEN, y todos los mensajes colocados utilizando este descriptor de contexto de cola se envían a dicha instancia.
  - Si se especifica OOBNDN, el gestor de colas puede elegir una instancia diferente de la cola de destino (que reside en un gestor de colas diferente del clúster) para cada llamada MQPUT sucesiva que utilice este descriptor de contexto de cola.

  Si la aplicación necesita enviar un mensaje a una instancia *específica* de una cola de clúster (es decir, una instancia de cola que reside en un gestor de colas determinado del clúster), la aplicación debe especificar el nombre de dicho gestor de colas en el campo *ODMN* . Esto fuerza al gestor de colas local a enviar el mensaje al gestor de colas de destino especificado.
- Si el objeto que se está abriendo es una lista de distribución (es decir, *ODREC* es mayor que cero), *ODMN* debe estar en blanco o la serie nula. Si no se cumple esta condición, la llamada falla con el código de razón RC2153.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ODON* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona LNQMN. El valor inicial de este campo es de 48 caracteres en blanco.

ODON (serie de caracteres de 48 bytes)

El nombre del objeto.

Es el nombre local del objeto tal como se define en el gestor de colas identificado por *ODMN*. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (A-Z)
- Caracteres alfabéticos en minúsculas (a-z)
- Dígitos numéricos (0-9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Se puede utilizar un carácter nulo para indicar el final de datos significativos en el nombre; el nulo y cualquier carácter que lo siga se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ODON* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ODON* el nombre de la cola creada. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si el objeto que se está abriendo es una lista de distribución (es decir, *ODREC* está presente y es mayor que cero), *ODON* debe estar en blanco o la serie nula. Si no se cumple esta condición, la llamada falla con el código de razón RC2152.
- Si *ODOT* es OTQM, se aplican reglas especiales; en este caso, el nombre debe estar completamente en blanco hasta el primer carácter nulo o el final del campo.
- Si ODON es el nombre de una cola alias con TARGTYPE (TOPIC), primero se realiza una comprobación de seguridad en la cola alias con nombre, como es normal para el uso de colas alias. Si esta comprobación de seguridad es satisfactoria, esta llamada MQOPEN continuará y se comportará como una MQOPEN de una OTTOP, incluida la realización de una comprobación de seguridad en el objeto de tema administrativo.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ODON* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ODON* y *ODOS*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

ODORO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de objeto desde el inicio de MQOD.

Es el desplazamiento en bytes del primer registro de objeto MQOR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ODORO* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *ODREC* es cero.

Cuando se está abriendo una lista de distribución, se debe proporcionar una matriz de uno o más registros de objeto MQOR para especificar los nombres de las colas de destino en la lista de distribución. Esto se puede hacer de una de dos maneras:

- Utilizando el campo de desplazamiento *ODORO*

  En este caso, la aplicación debe declarar su propia estructura que contenga un MQOD seguido de la matriz de registros MQOR (con tantos elementos de matriz como sean necesarios) y establecer *ODORO* en el desplazamiento del primer elemento de la matriz desde el inicio del MQOD. Se debe tener cuidado para asegurarse de que este desplazamiento es correcto.

- Utilizando el campo de puntero *ODORP*

  En este caso, la aplicación puede declarar la matriz de estructuras MQOR por separado de la estructura MQOD y establecer *ODORP* en la dirección de la matriz.

Sea cual sea la técnica elegida, se debe utilizar uno de *ODORO* y *ODORP* ; la llamada falla con el código de razón RC2155 si ambos son cero, o ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODORP (puntero)

Dirección del primer registro de objeto.

Es la dirección del primer registro de objeto MQOR. *ODORP* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *ODREC* es cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Se puede utilizar *ODORP* o *ODORO* para especificar los registros de objeto, pero no ambos; consulte la descripción del campo *ODORO* anteriormente para obtener más detalles. Si no se utiliza *ODORP* , debe establecerse en el puntero nulo o en bytes nulos. Este campo se ignora si *ODVER* es menor que ODVER2.

ODOS (MQCHARV)

ODOS especifica el nombre de objeto largo que se va a utilizar.

Sólo se hace referencia a este campo para determinados valores de *ODOT*. Consulte la descripción de *ODOT* para obtener detalles de qué valores indican que se utiliza este campo.

Si *ODOS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2441.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ODON* y *ODOS*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema. Este campo se ignora si *ODVER* es menor que ODVER4.

ODOT (entero con signo de 10 dígitos)

Tipo de objeto.

Tipo de objeto que se denomina en *ODON*. Los valores posibles son:

OTQ
: Cola. El nombre del objeto se encuentra en *ODON*.

OTNLST
: Lista de nombres. El nombre del objeto se encuentra en *ODON*.

OPRO
: . El nombre del objeto se encuentra en *ODON*.

OTQM
: Gestor de colas. El nombre del objeto se encuentra en *ODON*.

OTTOP
: . El nombre completo del tema se puede crear a partir de dos campos diferentes: *ODON* y *ODOS*.

  Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

  Si no se puede encontrar el objeto identificado por el campo *ODON* , la llamada fallará con el código de razón RC2425 incluso si hay una serie especificada en *ODOS*.

Siempre es un campo de entrada. El valor inicial de este campo es OTQ.

ODREC (entero con signo de 10 dígitos)

Número de registros de objeto presentes.

Es el número de registros de objeto MQOR que ha proporcionado la aplicación. Si este número es mayor que cero, indica que se está abriendo una lista de distribución, siendo *ODREC* el número de colas de destino de la lista. Es válido que una lista de distribución contenga sólo un destino.

El valor de *ODREC* no debe ser menor que cero, y si es mayor que cero *ODOT* debe ser OTQ; la llamada falla con el código de razón RC2154 si no se cumplen estas condiciones.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODRMN (serie de caracteres de 48 bytes)

Nombre de gestor de colas resuelto.

Es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *ODRQN*. *ODRMN* puede ser el nombre del gestor de colas local.

Si *ODRQN* es una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ODRMN* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *ODRQN* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ODRMN* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una cola de clúster con OOBNDN especificado (o con OOBNDQ en vigor cuando el atributo de cola **DefBind** tiene el valor BNDNOT)
- Una lista de distribución

Se trata de un campo de salida. La longitud de este campo la proporciona LNQN. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *ODVER* es menor que ODVER3.

ODRO (MQCHARV)

ODRO es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en *ODON*.

Este campo sólo se devuelve para determinados tipos de objetos, temas y alias de cola que hacen referencia a un objeto de tema.

Si el nombre de objeto largo se proporciona en *ODOS* y no se proporciona nada en *ODON*, el valor devuelto en este campo es el mismo que el proporcionado en *ODOS*.

Si se omite este campo (es decir, ODRO.VSBufSize es cero), no se devuelve *ODRO* , pero la longitud se devuelve en ODRO.VSLength. Si la longitud es más corta que la *ODRO* completa, se trunca y devuelve tantos de los caracteres situados más a la derecha como caben en la longitud proporcionada.

Si *ODRO* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2520. Este campo se ignora si *ODVER* es menor que ODVER4.

ODRQN (serie de caracteres de 48 bytes)

Nombre de cola resuelto.

Es el nombre de la cola de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ODRMN*.

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ODRQN* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una lista de distribución
- Una cola alias que hace referencia a un objeto de tema (consulte "ODRO (MQCHARV)" en la página 1191 en su lugar)

Se trata de un campo de salida. La longitud de este campo la proporciona LNQN. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *ODVER* es menor que ODVER3.

ODRRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de respuesta desde el inicio de MQOD.

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ODRRO* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *ODREC* es cero.

Cuando se está abriendo una lista de distribución, se puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas que no se han podido abrir (campo*RRCC* en MQRR) y la razón de cada anomalía (campo*RRREA* en MQRR). Los datos se devuelven en la matriz de registros de respuesta en el mismo orden en que aparecen los nombres de cola en la matriz de registros de objeto. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunas colas se han abierto correctamente mientras que otras han fallado, o todas han fallado pero por motivos diferentes); el código de razón RC2136 de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro **REASON** de la llamada MQOPEN o MQPUT1 y los registros de respuesta no se establecen. Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *ODREC* de ellos.

Los registros de respuesta se pueden proporcionar de la misma forma que los registros de objeto, ya sea especificando un desplazamiento en *ODRRO*, o especificando una dirección en *ODRRP* ; consulte la descripción de *ODORO* anteriormente para obtener detalles sobre cómo hacerlo. Sin embargo, no se puede utilizar más de uno de *ODRRO* y *ODRRP* ; la llamada falla con el código de razón RC2156 si ambos son distintos de cero.

Para la llamada MQPUT1 , estos registros de respuesta se utilizan para devolver información sobre los errores que se producen cuando se envía el mensaje a las colas de la lista de distribución, así como los errores que se producen cuando se abren las colas. El código de terminación y el código de razón de la operación de transferencia para una cola sustituyen a los de la operación abierta para dicha cola sólo si el código de terminación de esta última era CCOK o CCWARN.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODRRP (puntero)

Dirección del primer registro de respuesta.

Es la dirección del primer registro de respuesta MQRR. *ODRRP* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *ODREC* es cero.

Se puede utilizar *ODRRP* o *ODRRO* para especificar los registros de respuesta, pero no ambos; consulte la descripción anterior del campo *ODRRO* para obtener más detalles. Si no se utiliza *ODRRP* , debe establecerse en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *ODVER* es menor que ODVER2.

ODSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

ODSIDV
Identificador de la estructura de descriptor de objeto.

Siempre es un campo de entrada. El valor inicial de este campo es ODSIDV.

ODSS (MQCHARV)

ODSS contiene la serie utilizada para proporcionar los criterios de selección utilizados al recuperar mensajes de una cola.

ODSS no debe proporcionarse en los casos siguientes:

- Si *ODOT* no es OTQ
- Si la cola que se está abriendo no se está abriendo utilizando una de las opciones de entrada, OOINP*

Si se proporciona *ODSS* en estos casos, la llamada falla con el código de razón RC2516.

Si *ODSS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2519. Este campo se ignora si *ODVER* es menor que ODVER4.

ODUDC (entero con signo de 10 dígitos)

Número de colas remotas abiertas satisfactoriamente.

Es el número de colas de la lista de distribución que se resuelven en colas remotas y que se han abierto correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

ODVER1
Estructura del descriptor de objeto Version-1 .

ODVER2
Estructura del descriptor de objeto Version-2 .

ODVER3
Estructura del descriptor de objeto Version-3 .

ODVER4
Estructura del descriptor de objeto Version-4 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

ODVERC
Versión actual de la estructura de descriptor de objeto.

Siempre es un campo de entrada. El valor inicial de este campo es ODVER1.

Valores iniciales

Tabla 713. Valores iniciales de campos en MQOD

Nombre de campo	Nombre de constante	Valor de constante
ODSID	ODSIDV	'OD¬¬'

Tabla 713. Valores iniciales de campos en MQOD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
ODVER	ODVER1	1
ODOT	OTQ	1
ODON	Ninguna	Espacios en blanco
ODMN	Ninguna	Espacios en blanco
ODDN	Ninguna	`'AMQ.*'`
ODAU	Ninguna	Espacios en blanco
ODREC	Ninguna	0
ODKDC	Ninguna	0
ODUDC	Ninguna	0
ODIDC	Ninguna	0
ODORO	Ninguna	0
ODRRO	Ninguna	0
ODORP	Ninguna	Puntero nulo o bytes nulos
ODRRP	Ninguna	Puntero nulo o bytes nulos
ODASI	SINONA	Nulos
ODRQN	Ninguna	Espacios en blanco
ODRMN	Ninguna	Espacios en blanco
ODOS	Tal como se ha definido para MQCHARV	Tal como se ha definido para MQCHARV
ODRO	Tal como se proporciona en *ODOS*	Tal como se proporciona en *ODOS*
ODSS	Ninguna	Espacios en blanco
Notas: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID                  1      4    INZ('OD  ')
D*
D* Structure version number
D  ODVER                  5      8I 0 INZ(1)
D*
D* Object type
D  ODOT                   9     12I 0 INZ(1)
D*
D* Object name
D  ODON                  13     60    INZ
D*
D* Object queue manager name
D  ODMN                  61    108    INZ
```

```
D*
D* Dynamic queue name
D  ODDN                 109    156    INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU                 157    168    INZ
D*
** Number of object records
D* present
D  ODREC                169    172I 0 INZ(0)
D*
** Number of local queues opened
D* successfully
D  ODKDC                173    176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC                177    180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC                181    184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO                185    188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO                189    192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP                193    208*   INZ(*NULL)
D*
** Address of first response
D* record
D  ODRRP                209    224*   INZ(*NULL)
D*
D* Alternate security identifier
D  ODASI                225    264    INZ(X'00000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     0000000000000')
D*
D* Resolved queue name
D  ODRQN                265    312    INZ
D*
D* Resolved queue manager name
D  ODRMN                313    360    INZ
D*
D* reserved field
D  ODRE1                361    364I 0 INZ(0)
D*
D* reserved field
D  ODRS2                365    368I 0 INZ(0)
D*
D* Object long name
D* Address of variable length string
D  ODOSCHRP             369    384*   INZ(*NULL)
D* Offset of variable length string
D  ODOSCHRO             385    388I 0 INZ(0)
D* Size of buffer
D  ODOSVSBS             389    392I 0 INZ(-1)
D* Length of variable length string
D  ODOSCHRL             393    396I 0 INZ(0)
D* CCSID of variable length string
D  ODOSCHRC             397    400I 0 INZ(-3)
D*
D* Message Selector
D* Address of variable length string
D  ODSSCHRP             401    416*   INZ(*NULL)
D* Offset of variable length string
D  ODSSCHRO             417    420I 0 INZ(0)
D* Size of buffer
D  ODSSVSBS             421    424I 0 INZ(-1)
D* Length of variable length string
D  ODSSCHRL             425    428I 0 INZ(0)
D* CCSID of variable length string
D  ODSSCHRC             429    432I 0 INZ(-3)
D*
D* Resolved long object name
D* Address of variable length string
```

```
     D  ODRSOCHRP            433    448*   INZ(*NULL)
     D* Offset of variable length string
     D  ODRSOCHRO            449    452I 0 INZ(0)
     D* Size of buffer
     D  ODRSOVSBS            453    456I 0 INZ(-1)
     D* Length of variable length string
     D  ODRSOCHRL            457    460I 0 INZ(0)
     D* CCSID of variable length string
     D  ODRSOCHRC            461    464I 0 INZ(-3)
     D*
     D* Alias queue resolved object type
     D  ODRT                 465    468I 0 INZ(0)
```

IBM i MQOR (registro de objeto) en IBM i

La estructura MQOR se utiliza para especificar el nombre de cola y el nombre de gestor de colas de una única cola de destino.

Visión general

Finalidad: MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1 .

Conjunto de caracteres y codificación: Los datos de MQOR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en la llamada MQOPEN, es posible abrir una lista de colas; esta lista se denomina *lista de distribución*. Cada mensaje colocado utilizando el descriptor de contexto de cola devuelto por esa llamada MQOPEN se coloca en cada una de las colas de la lista, si la cola se ha abierto correctamente.

- "Campos" en la página 1196
- "Valores iniciales" en la página 1197
- "Declaración RPG" en la página 1197

Campos

La estructura MQOR contiene los campos siguientes; los campos se describen en **orden alfabético**:

ORMN (serie de caracteres de 48 bytes)

Nombre del gestor de colas de objetos.

Es el mismo que el campo *ODMN* en la estructura MQOD (consulte MQOD para obtener más detalles).

Siempre es un campo de entrada. El valor inicial de este campo es de 48 caracteres en blanco.

ORON (serie de caracteres de 48 bytes)

El nombre del objeto.

Es el mismo que el campo *ODON* en la estructura MQOD (consulte MQOD para obtener detalles), excepto que:

- Debe ser el nombre de una cola.
- No debe ser el nombre de una cola modelo.

Siempre es un campo de entrada. El valor inicial de este campo es de 48 caracteres en blanco.

Valores iniciales

Tabla 714. Valores iniciales de los campos en MQOR		
Nombre de campo	**Nombre de constante**	**Valor de constante**
ORON	Ninguna	Espacios en blanco
ORMN	Ninguna	Espacios en blanco

Declaración 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-Descriptor de propiedad

MQPD se utiliza para definir los atributos de una propiedad.

Visión general

Finalidad: la estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Conjunto de caracteres y codificación: los datos de MQPD deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- "Campos" en la página 1197
- "Valores iniciales" en la página 1200
- "Declaración RPG" en la página 1200

Campos

La estructura MQPD contiene los campos siguientes; los campos se describen en **orden alfabético**:

PDCT (entero con signo de 10 dígitos)

Describe a qué contexto de mensaje pertenece la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por IBM MQque el gestor de colas reconoce como incorrecta. el gestor de colas corrige el valor del campo *PDCT* .

Se puede especificar la opción siguiente:

PDUSC
La propiedad está asociada al contexto de usuario.

No se requiere ninguna autorización especial para poder establecer una propiedad asociada al contexto de usuario utilizando la llamada MQSETMP.

En un gestor de colas de IBM WebSphere MQ 7.0 , una propiedad asociada con el contexto de usuario se guarda tal como se describe para OOSAVA. Una llamada MQPUT con PMPASA especificado, hace que la propiedad se copie del contexto guardado en el nuevo mensaje.

Si la opción descrita anteriormente no es necesaria, se puede utilizar la opción siguiente:

PNOC
La propiedad no está asociada a un contexto de mensaje.

Se rechaza un valor no reconocido con un código *PDREA* de RC2482.

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es PDNOC.

PDCPYOPT (entero con signo de 10 dígitos)

Describe en qué tipo de mensajes se debe copiar la propiedad.

Es un campo de sólo salida para las propiedades definidas por IBM MQreconocidas; IBM MQ establece el valor adecuado.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por IBM MQque el gestor de colas reconoce como incorrecta. el gestor de colas corrige el valor del campo `CopyOptions` .

Puede especificar una o varias de estas opciones. Para especificar más de una opción, añada los valores juntos (no añada la misma constante más de una vez) o combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

COPFOR
Esta propiedad se copia en un mensaje que se está reenviando.

COPPUB
Esta propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

COPREP
Esta propiedad se copia en un mensaje de respuesta.

COPRP
Esta propiedad se copia en un mensaje de informe.

COPIAR
Esta propiedad se copia en todos los tipos de mensajes posteriores.

COPNÓN
Esta propiedad no se copia en un mensaje.

Opción predeterminada: Se puede especificar la siguiente opción para proporcionar el conjunto predeterminado de opciones de copia:

COPDEF
Esta propiedad se copia en un mensaje que se está reenviando, en un mensaje de informe o en un mensaje recibido por un suscriptor cuando se está publicando un mensaje.

Esto equivale a especificar la combinación de opciones COPFOR, más COPRP, más COPPUB.

Si no se requiere ninguna de las opciones descritas anteriormente, utilice la opción siguiente:

COPNÓN
Utilice este valor para indicar que no se ha especificado ninguna otra opción de copia; programáticamente no existe ninguna relación entre esta propiedad y los mensajes posteriores. Esto siempre se devuelve para las propiedades del descriptor de mensaje.

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es COPDEF.

PDOPT (entero con signo de 10 dígitos)

El valor debe ser:

PNONE
No se ha especificado ninguna opción

Siempre es un campo de entrada. El valor inicial de este campo es PDNONE.

PDSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

PIDV
Identificador de la estructura del descriptor de propiedad.

Siempre es un campo de entrada. El valor inicial de este campo es **PSIDV**.

PDSUP (entero con signo de 10 dígitos)

Este campo describe qué nivel de soporte para la propiedad de mensaje es necesario para el gestor de colas, para que el mensaje que contiene esta propiedad se coloque en una cola. Esto sólo se aplica a las propiedades definidas por IBM MQ; el soporte para todas las demás propiedades es opcional.

El campo se establece automáticamente en el valor correcto cuando el gestor de colas conoce la propiedad definida por IBM MQ. Si la propiedad no se reconoce, se asigna PDSUPO. Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por IBM MQque el gestor de colas reconoce como incorrecta. el gestor de colas corrige el valor del campo *PDSUP* .

Cuando se establece una propiedad definida por IBM MQutilizando la llamada MQSETMP en un descriptor de mensaje donde se ha establecido la opción CMNOVA, *PDSUP* se convierte en un campo de entrada. Esto permite a una aplicación colocar una propiedad definida por IBM MQ, con el valor correcto, donde la propiedad no está soportada por el gestor de colas conectado, pero donde el mensaje está pensado para procesarse en otro gestor de colas.

El valor PDSUPO siempre se asigna a propiedades que no son propiedades definidas por IBM MQ.

Si un gestor de colas IBM WebSphere MQ 7.0 , que da soporte a las propiedades de mensaje, recibe una propiedad que contiene un valor *PDSUP* no reconocido, la propiedad se trata como si:

- Se ha especificado PDSUPR si alguno de los valores no reconocidos están contenidos en el PDRUM.
- Se ha especificado PDSUPL si alguno de los valores no reconocidos están contenidos en el PDAUXM
- De lo contrario, se ha especificado PDSUPO.

La llamada MQINQMP devuelve uno de los valores siguientes, o se puede especificar uno de los valores, cuando se utiliza la llamada MQSETMP en un descriptor de mensaje en el que se ha establecido la opción CMNOVA:

PDSUPO
Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas por IBM MQ.

PDSUPR
Se necesita soporte para la propiedad. El mensaje es rechazado por un gestor de colas que no da soporte a la propiedad definida por IBM MQ. La llamada MQPUT o MQPUT1 falla con el código de terminación CCFAIL y el código de razón RC2490.

PSUPL
El mensaje es rechazado por un gestor de colas que no soporta la propiedad definida por IBM MQsi el mensaje está destinado a una cola local. La llamada MQPUT o MQPUT1 falla con el código de terminación CCFAIL y el código de razón RC2490.

La llamada MQPUT o MQPUT1 se ejecuta correctamente si el mensaje está destinado a un gestor de colas remoto.

Es un campo de salida en la llamada MQINQMP y un campo de entrada en la llamada MQSETMP si el descriptor de mensaje se ha creado con la opción CMNOVA establecida. El valor inicial de este campo es PDSUPO.

PDVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

PDVER1
Estructura del descriptor de propiedad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

PDVERC
Versión actual de la estructura del descriptor de propiedad.

Siempre es un campo de entrada. El valor inicial de este campo es **PDVER1**.

Valores iniciales

Tabla 715. Valores iniciales de los campos en MQPD

Nombre de campo	Nombre de constante	Valor de constante
PDSID	PDSIDV	'PD'
PDVER	PDVER1	1
PDOPT	PNONE	0
PDSUP	PDSUPO	0
PDCT	PNOC	0
PDCPYOPT	COPDEF	0

Declaración 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 (opciones de transferencia de mensajes) en IBM i

La estructura MQPMO permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o se publican en los temas.

Visión general

Finalidad
La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1 .

Versión
La versión actual de MQPMO es PMVER2. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

El archivo COPY proporcionado contiene la versión más reciente de MQPMO soportada por el entorno, pero con el valor inicial del campo *PMVER* establecido en PMVER1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *PMVER* en el número de versión de la versión necesaria.

Juego de caracteres y codificación
Los datos de MQPMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la página 1201
- “Valores iniciales” en la página 1215
- “Declaración RPG” en la página 1215

Campos

La estructura MQPMO contiene los campos siguientes; los campos se describen en orden alfabético:

PMCT (entero con signo de 10 dígitos)

Descriptor de contexto de objeto de la cola de entrada.

Si se especifica PMPASI o PMPASA, este campo debe contener el manejador de cola de entrada del que se toma la información de contexto que se va a asociar con el mensaje que se va a colocar.

Si no se especifican PMPASI y PMPASA, este campo se ignora.

Este es un campo de entrada. El valor inicial de este campo es `0`.

PMIDC (entero con signo de 10 dígitos)

Número de mensajes que no se han podido enviar.

Es el número de mensajes que no se han podido enviar a las colas de la lista de distribución. El recuento incluye las colas que no se han podido abrir y las colas que se han abierto correctamente pero para las que la operación de colocación ha fallado. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Nota: Este campo sólo se establece si el parámetro **CMPCOD** de la llamada MQPUT o MQPUT1 es CCOK o CCWARN; no se establece si el parámetro **CMPCOD** es CCFAIL.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que PMVER2.

PMKDC (entero con signo de 10 dígitos)

Número de mensajes enviados satisfactoriamente a colas locales.

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que son colas locales. El recuento no incluye los mensajes enviados a colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que PMVER2.

PMOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQPUT y MQPUT1.

Se puede especificar cualquiera o ninguno de los siguientes. Si se necesita más de uno, se pueden añadir los valores (no añadir la misma constante más de una vez). Las combinaciones que no son válidas se anotan; cualquier otra combinación es válida.

Opciones de publicación: Las opciones siguientes controlan la forma en que se publican los mensajes en un tema.

PMSRTO

Cualquier información rellenada en los campos MDRQ y MDRM del MQMD de esta publicación no se pasa a los suscriptores. Si esta opción se utiliza con una opción de informe que requiere una cola ReplyTo, la llamada falla con `RC2027` .

PRET

El gestor de colas debe retener la publicación que se está enviando. Esto permite a un suscriptor solicitar una copia de esta publicación después de la hora en que se publicó, utilizando la llamada MQSUBRQ. También permite que una publicación se envíe a las aplicaciones que hacen su suscripción después de la hora en que se realizó esta publicación, a menos que opten por no enviarla utilizando la opción SONEWP. Si a una aplicación se le envía una publicación que se ha retenido, se indica mediante la propiedad de mensaje mq.IsRetained de dicha publicación.

Sólo se puede retener una publicación en cada nodo del árbol de temas. Esto significa que si ya hay una publicación retenida para este tema, publicada por cualquier otra aplicación, se sustituye por esta publicación. Por lo tanto, es mejor evitar que más de un editor retenga mensajes sobre el mismo tema.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en el tema, en cuyo caso pueden coincidir varias publicaciones retenidas (en varios nodos del árbol de temas) y se pueden enviar varias publicaciones a la aplicación solicitante. Consulte la descripción de la llamada "MQSUBRQ-Solicitud de suscripción" en la página 807 para obtener más detalles.

Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con RC2479 .

Opciones de punto de sincronismo: Las opciones siguientes están relacionadas con la participación de la llamada MQPUT o MQPUT1 dentro de una unidad de trabajo:

PPMSYP

Transfiera el mensaje con el control de punto de sincronismo.

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

Si no se especifica esta opción y PMNSYP, la solicitud de colocación no está dentro de una unidad de trabajo.

PMSYP no debe especificarse con PMNSYP.

PMNSYP

Transfiera el mensaje sin control de punto de sincronismo.

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje está disponible inmediatamente y no se puede suprimir restituir una unidad de trabajo.

Si no se especifica esta opción y PMSYP, la solicitud de colocación no está dentro de una unidad de trabajo.

PMNSYP no debe especificarse con PMSYP.

Opciones de identificador de mensaje e identificador de correlación: las opciones siguientes solicitan al gestor de colas que genere un nuevo identificador de mensaje o identificador de correlación:

PMNMID

Generar un nuevo identificador de mensaje.

Esta opción hace que el gestor de colas sustituya el contenido del campo *MDMID* en MQMD por un nuevo identificador de mensaje. Este identificador de mensaje se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

Esta opción también se puede especificar cuando el mensaje se transfiere a una lista de distribución; consulte la descripción del campo *PRMID* en la estructura MQPMR para obtener más detalles.

El uso de esta opción libera a la aplicación de la necesidad de restablecer el campo *MDMID* en MINONE antes de cada llamada MQPUT o MQPUT1 .

PNCID

Generar un nuevo identificador de correlación.

Esta opción hace que el gestor de colas sustituya el contenido del campo *MDCID* en MQMD por un nuevo identificador de correlación. Este identificador de correlación se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

Esta opción también se puede especificar cuando el mensaje se transfiere a una lista de distribución; consulte la descripción del campo *PRCID* en la estructura MQPMR para obtener más detalles.

PMNCID es útil en situaciones en las que la aplicación requiere un identificador de correlación exclusivo.

Opciones de grupo y segmento: la opción siguiente está relacionada con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Estas definiciones pueden ser de ayuda para comprender la opción:

Mensaje físico

Esta es la unidad de información más pequeña que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Generalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (campo*MDMID* en MQMD), aunque el gestor de colas no los impone.

Mensaje lógico

Se trata de una única unidad de información de aplicación. En ausencia de restricciones del sistema, un mensaje lógico sería el mismo que un mensaje físico. Pero cuando los mensajes lógicos son grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados *segmentos*.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (campo*MDGID* en MQMD) y el mismo número de secuencia de mensaje (campo*MDSEQ* en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento (campo*MDOFF* en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Puesto que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo, aunque en este caso sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos para los que la aplicación emisora ha inhibido la segmentación tienen un identificador de grupo nulo (GINONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes para el número de secuencia de mensaje, que es un entero en el rango de 1 a n, donde n es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

PLOGO

Los mensajes en grupos y segmentos de mensajes lógicos se colocan en orden lógico.

Esta opción indica al gestor de colas cómo la aplicación coloca los mensajes en grupos y segmentos de mensajes lógicos. Sólo se puede especificar en la llamada MQPUT; no es válida en la llamada MQPUT1.

Si se especifica PMLOGO, indica que la aplicación utiliza llamadas MQPUT sucesivas para:

- Poner los segmentos de cada mensaje lógico en el orden de desplazamiento de segmento creciente, empezando desde 0, sin huecos.

- Coloque todos los segmentos en un mensaje lógico antes de colocar los segmentos en el siguiente mensaje lógico.
- Poner los mensajes lógicos de cada grupo de mensajes en el orden de número de secuencia de mensaje creciente, empezando desde 1, sin huecos.
- Coloque todos los mensajes lógicos en un grupo de mensajes antes de colocar los mensajes lógicos en el siguiente grupo de mensajes.

Este orden se denomina "orden lógico".

Puesto que la aplicación ha indicado al gestor de colas cómo coloca los mensajes en grupos y segmentos de mensajes lógicos, la aplicación no tiene que mantener y actualizar la información de grupo y segmento sobre cada llamada MQPUT, ya que el gestor de colas lo hace. En concreto, significa que la aplicación no necesita establecer los campos *MDGID*, *MDSEQ*y *MDOFF* en MQMD, ya que el gestor de colas los establece en los valores adecuados. La aplicación sólo necesita establecer el campo *MDMFL* en MQMD, para indicar cuándo los mensajes pertenecen a grupos o son segmentos de mensajes lógicos, y para indicar el último mensaje de un grupo o último segmento de un mensaje lógico.

Una vez que se ha iniciado un grupo de mensajes o un mensaje lógico, las llamadas MQPUT posteriores deben especificar los distintivos MF* adecuados en *MDMFL* en MQMD. Si la aplicación intenta colocar un mensaje que no está en un grupo cuando hay un grupo de mensajes sin terminar, o colocar un mensaje que no es un segmento cuando hay un mensaje lógico sin terminar, la llamada falla con el código de razón RC2241 o RC2242 , según corresponda. Sin embargo, el gestor de colas conserva la información sobre el grupo de mensajes actual o el mensaje lógico actual, y la aplicación puede terminarlos enviando un mensaje (posiblemente sin datos de mensaje de aplicación) especificando MFLMIG o MFLSEG según corresponda, antes de volver a emitir la llamada MQPUT para colocar el mensaje que no está en el grupo o no es un segmento.

La Tabla 716 en la página 1205 muestra las combinaciones de opciones y distintivos que son válidos, y los valores de los campos *MDGID*, *MDSEQ*y *MDOFF* que el gestor de colas utiliza en cada caso. Las combinaciones de opciones y distintivos que no aparecen en la tabla no son válidas. Las columnas de la tabla tienen los significados siguientes:

LOG ORD
Indica si se ha especificado la opción PMLOGO en la llamada.

MIG
Indica si se ha especificado la opción MFMIG o MFLMIG en la llamada.

SEG
Indica si se ha especificado la opción MFSEG o MFLSEG en la llamada.

SEG OK
Indica si se ha especificado la opción MFSEGA en la llamada.

Cur grp
Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg
Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas
Muestra los valores que utiliza el gestor de colas. "Anterior" indica el valor utilizado para el campo en el mensaje anterior para el descriptor de contexto de cola.

PMRLOC
Especifica que el PMRQN de la estructura MQPMO debe completarse con el nombre de la cola local a la que se coloca realmente el mensaje. El nombre ResolvedQMgrse completa de forma similar con el nombre del gestor de colas local que aloja la cola local. Consulte OORLOQ para ver qué significa esto. Si un usuario está autorizado para una colocación en una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQPUT. No se necesita ninguna autorización especial.

Tabla 716. Opciones de MQPUT relacionadas con los mensajes de grupos y los segmentos de mensajes lógicos

Opciones especificadas				**Estado de grupo y mensaje lógico antes de la llamada**		**Valores que utiliza el gestor de colas**		
LOG ORD	**MIG**	**SEG**	**SEG OK**	**Cur grp**	**Cur log msg**	*MDGID*	*MDSEQ*	*MDOFF*
Sí	No	No	No	No	No	GINONA	1	0
Sí	No	No	Sí	No	No	Nuevo ID de grupo	1	0
Sí	No	Sí	Sí o no	No	No	Nuevo ID de grupo	1	0
Sí	No	Sí	Sí o no	No	Sí	ID de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior
Sí	Sí	Sí o no	Sí o no	No	No	Nuevo ID de grupo	1	0
Sí	Sí	Sí o no	Sí o no	Sí	No	ID de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Sí o no	Sí	Sí	ID de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	No	No	No	Sí o no	Sí o no	GINONA	1	0
No	No	No	Sí	Sí o no	Sí o no	Nuevo ID de grupo si GINONE, de lo contrario valor en el campo	1	0
No	No	Sí	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si GINONE, de lo contrario valor en el campo	1	Valor en campo
No	Sí	No	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si GINONE, de lo contrario valor en el campo	Valor en campo	0
No	Sí	Sí	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si GINONE, de lo contrario valor en el campo	Valor en campo	Valor en campo

Nota:

- PMLOGO no es válido en la llamada MQPUT1 .
- Para el campo *MDMID* , el gestor de colas genera un nuevo identificador de mensaje si se especifica PMNMID o MINONE y, de lo contrario, utiliza el valor del campo.
- Para el campo *MDCID* , el gestor de colas genera un nuevo identificador de correlación si se especifica PMNCID y, de lo contrario, utiliza el valor del campo.

Cuando se especifica PMLOGO, el gestor de colas requiere que todos los mensajes de un grupo y segmentos de un mensaje lógico se pongan con el mismo valor en el campo *MDPER* de MQMD, es decir, todos deben ser persistentes, o todos deben ser no persistentes. Si esta condición no se cumple, la llamada MQPUT falla con el código de razón RC2185 .

La opción PMLOGO afecta a las unidades de trabajo como se indica a continuación:

- Si el primer mensaje físico de un grupo o mensaje lógico se coloca dentro de una unidad de trabajo, todos los demás mensajes físicos del grupo o mensaje lógico deben colocarse dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola. Sin embargo, no es necesario que se coloque dentro de la misma unidad de trabajo. Esto permite que un grupo de mensajes o mensaje lógico que consta de muchos mensajes físicos se divida en dos o más unidades de trabajo consecutivas para el descriptor de contexto de cola.
- Si el primer mensaje físico de un grupo o mensaje lógico no se coloca dentro de una unidad de trabajo, ninguno de los demás mensajes físicos del grupo o mensaje lógico se puede colocar dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola.

Si no se cumplen estas condiciones, la llamada MQPUT falla con el código de razón RC2245 .

Cuando se especifica PMLOGO, el MQMD proporcionado en la llamada MQPUT no debe ser menor que MDVER2. Si esta condición no se cumple, la llamada falla con el código de razón RC2257 .

Si no se especifica PMLOGO, los mensajes en grupos y segmentos de mensajes lógicos se pueden poner en cualquier orden, y no es necesario colocar grupos de mensajes completos o mensajes lógicos completos. Es responsabilidad de la aplicación asegurarse de que los campos *MDGID*, *MDSEQ*, *MDOFF*y *MDMFL* tienen los valores adecuados.

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico en el medio, después de que se haya producido una anomalía del sistema. Cuando se reinicia el sistema, la aplicación puede establecer los campos *MDGID*, *MDSEQ*, *MDOFF*, *MDMFL*y *MDPER* en los valores adecuados y, a continuación, emitir la llamada MQPUT con PMSYP o PMNSYP establecido como *necesario*, pero sin especificar PMLOGO. Si esta llamada es satisfactoria, el gestor de colas conserva la información de grupo y segmento, y las llamadas MQPUT posteriores que utilizan ese descriptor de contexto de cola pueden especificar PMLOGO como normal.

La información de grupo y segmento que el gestor de colas retiene para la llamada MQPUT está separada de la información de grupo y segmento que retiene para la llamada MQGET.

Para cualquier descriptor de contexto de cola determinado, la aplicación es libre de mezclar llamadas MQPUT que especifiquen PMLOGO con llamadas MQPUT que no lo hagan, pero se deben tener en cuenta los puntos siguientes:

- Si no se especifica PMLOGO, cada llamada MQPUT satisfactoria hace que el gestor de colas establezca la información de grupo y segmento para el descriptor de contexto de cola en los valores especificados por la aplicación; esto sustituye la información de grupo y segmento existente retenida por el gestor de colas para el descriptor de contexto de cola.
- Si no se especifica PMLOGO, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; sin embargo, la llamada podría tener éxito con un código de terminación CCWARN. La Tabla 717 en la página 1207 muestra los distintos casos que se pueden presentar. En estos casos, si el código de terminación no es CCOK, el código de razón es uno de los siguientes (según corresponda):
  - RC2241
  - RC2242
  - RC2185
  - RC2245

  Nota: El gestor de colas no comprueba la información de grupo y de segmento para la llamada MQPUT1.

Tabla 717. Resultado cuando la llamada MQPUT o MQCLOSE no es coherente con la información de grupo y segmento		
La llamada actual es	**La llamada anterior era MQPUT con PMLOGO**	**La llamada anterior era MQPUT sin PMLOGO**
MQPUT con PMLOGO	CCFAIL	CCFAIL
MQPUT sin PMLOGO	CCWARN	CCOK
MQCLOSE con un grupo o mensaje lógico sin terminar	CCWARN	CCOK

Las aplicaciones que simplemente desean poner mensajes y segmentos en orden lógico se recomiendan para especificar PMLOGO, ya que esta es la opción más sencilla de utilizar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, las aplicaciones especializadas pueden necesitar más control que el proporcionado por la opción PMLOGO, y esto se puede lograr no especificando esa opción. Si se hace esto, la aplicación debe asegurarse de que los campos *MDGID*, *MDSEQ*, *MDOFF*y *MDMFL* en MQMD se hayan establecido correctamente, antes de cada llamada MQPUT o MQPUT1 .

Por ejemplo, una aplicación que desea reenviar mensajes físicos que recibe, sin tener en cuenta si esos mensajes están en grupos o segmentos de mensajes lógicos, no debe especificar PMLOGO. Existen dos razones para ello:

- Si los mensajes se recuperan y se ponen en orden, la especificación de PMLOGO hace que se asigne un nuevo identificador de grupo a los mensajes, lo que puede dificultar o hacer imposible que el originador de los mensajes correlacione los mensajes de respuesta o de informe que resulten del grupo de mensajes.
- En una red compleja con múltiples rutas entre los gestores de colas de emisión y recepción, los mensajes físicos pueden llegar fuera de secuencia. Al no especificar PMLOGO y el GMLOGO correspondiente en la llamada MQGET, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llegue, sin necesidad de esperar a que llegue el siguiente en orden lógico.

Las aplicaciones que generan mensajes de informe para mensajes en grupos o segmentos de mensajes lógicos tampoco deben especificar PMLOGO al colocar el mensaje de informe.

PMLOGO se puede especificar con cualquiera de las otras opciones PM*.

Opciones de contexto: Las opciones siguientes controlan el proceso del contexto de mensaje:

PNOC
No se debe asociar ningún contexto con el mensaje.

Tanto el contexto de identidad como el contexto de origen se establecen para indicar que no hay contexto. Esto significa que los campos de contexto en MQMD se establecen en:

- Espacios en blanco para campos de caracteres
- Nulos para campos de bytes
- Ceros para campos numéricos

PMDEFC
Utilizar contexto predeterminado.

El mensaje debe tener asociada la información de contexto predeterminada, tanto para la identidad como para el origen. El gestor de colas establece los campos de contexto en el descriptor de mensaje como se indica a continuación:

Tabla 718. Valores de información de contexto predeterminados para campos MQMD

Campo de MQMD	Valor utilizado
MDUID	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
MDACC	Determinado a partir del entorno si es posible; de lo contrario, establézcalo en ACNONE.
MDAID	Establecido en blancos.
MDPAT	Determinado a partir del entorno.
MDPAN	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
MDPD	Se establece en la fecha en la que se coloca el mensaje.
MDPT	Establézcalo en la hora en que se coloca el mensaje.
MDAOD	Establecido en blancos.

Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta es la acción predeterminada si no se especifica ninguna opción de contexto.

PMPASI
Pasar contexto de identidad desde un descriptor de contexto de cola de entrada.

El mensaje tiene que tener información de contexto asociada. El contexto de identidad se toma del descriptor de contexto de cola especificado en el campo *PMCT* . El gestor de colas genera la información de contexto de origen de la misma forma que para PMDEFC (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOPASI (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción OOPASI.

PMPASA
Pasar todo el contexto desde un descriptor de contexto de cola de entrada.

El mensaje tiene que tener información de contexto asociada. Tanto la identidad como el contexto de origen se toman del descriptor de contexto de cola especificado en el campo *PMCT* . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOPASA (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción OOPASA.

PMSETI
Establezca el contexto de identidad de la aplicación.

El mensaje tiene que tener información de contexto asociada. La aplicación especifica el contexto de identidad en la estructura MQMD. El gestor de colas genera la información de contexto de origen de la misma forma que para PMDEFC (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOSETI (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción OOSETI.

PMSETA

Establezca todo el contexto de la aplicación.

El mensaje tiene que tener información de contexto asociada. La aplicación especifica la identidad y el contexto de origen en la estructura MQMD. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOSETA. Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción OOSETA.

Sólo se puede especificar una de las opciones de contexto PM*. Si no se especifica ninguna de estas opciones, se presupone PMDEFC.

Tipos de respuesta de colocación. Las opciones siguientes controlan la respuesta devuelta a una llamada MQPUT o MQPUT1 . Sólo puede especificar una de estas opciones. Si no se especifican PMARES y PMSRES, se presupone PMRASQ o PMRAST.

PMARES

La opción PMARES solicita que se complete una operación MQPUT o MQPUT1 sin que la aplicación espere a que el gestor de colas complete la llamada. La utilización de esta opción puede mejorar el rendimiento de la mensajería, especialmente para las aplicaciones que utilizan enlaces de cliente. Una aplicación puede comprobar periódicamente, utilizando el verbo MQSTAT, si se ha producido un error durante cualquier llamada asíncrona anterior.

Con esta opción, solo se garantiza que se completen los campos siguientes en el MQMD;

- MDAID
- MDPAT
- MDPAN
- MDAOD

Además, si se especifica PMNMID o PMNCID como opciones, o ambos, el MDMID y el MDCID devueltos también se completan. (PMNMID se puede especificar implícitamente especificando un campo MDMID en blanco).

Sólo se completan los campos especificados anteriormente. Otra información que normalmente se devolvería en la estructura MQMD o MQPMO no está definida.

Al solicitar una respuesta de transferencia asíncrona para MQPUT o MQPUT1, un CMPCOD y REASON de CCOK y RCNONE no significa necesariamente que el mensaje se haya transferido correctamente a una cola. Al desarrollar una aplicación MQI que utiliza una respuesta de transferencia asíncrona y que requiere confirmación de que los mensajes se han colocado en una cola, debe comprobar los códigos CMPCOD y REASON de las operaciones de transferencia y también utilizar MQSTAT para consultar información de error asíncrono.

Aunque es posible que no se devuelva inmediatamente el éxito o el error de cada llamada MQPUT/MQPUT1 individual, el primer error que se ha producido bajo una llamada asíncrona se puede determinar en una coyuntura posterior a través de una llamada a MQSTAT.

Si un mensaje persistente bajo punto de sincronismo no se puede entregar utilizando la respuesta de transferencia asíncrona e intenta confirmar la transacción, la confirmación falla y la transacción se restituye con un código de terminación de CCFAIL y una razón de RC2003 . La aplicación puede realizar una llamada a MQSTAT para determinar la causa de una anomalía anterior de MQPUT o MQPUT1

PMSRES

La especificación de este valor para una opción put en la estructura MQPMO garantiza que la operación MQPUT o MQPUT1 se emita siempre de forma síncrona. Si la operación es satisfactoria, se completan todos los campos de MQMD y MQPMO. Se proporciona para garantizar una respuesta síncrona independientemente del valor de respuesta de colocación predeterminado definido en la cola o el objeto de tema.

PMRASQ
Si se especifica este valor para una llamada MQPUT, el tipo de respuesta put utilizado se toma del valor DEFPRESP especificado en la cola cuando la aplicación lo abrió. Si una aplicación cliente está conectada a un gestor de colas en un nivel anterior a IBM WebSphere MQ 7.0, se comporta como si se hubiera especificado PMSRES.

Si se especifica esta opción para una llamada MQPUT1 , no se utiliza el valor DEFPRESP de la definición de cola. Si la llamada MQPUT1 utiliza PMSYP se comporta como para PMARES, y si utiliza PMNSYP se comporta como para PMSRES.

PMRAST
Es un sinónimo de PMRASQ para su uso con objetos de tema.

Otras opciones: Las opciones siguientes controlan la comprobación de autorización y lo que sucede cuando el gestor de colas se desactiva temporalmente:

PMALTU
Validar con el identificador de usuario especificado.

Esto indica que el campo *ODAU* del parámetro **OBJDSC** de la llamada MQPUT1 contiene un identificador de usuario que se debe utilizar para validar la autorización para colocar mensajes en la cola. La llamada sólo puede realizarse correctamente si este *ODAU* tiene autorización para abrir la cola con las opciones especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación tiene autorización para hacerlo. (Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.)

Esta opción sólo es válida con la llamada MQPUT1 .

PFIQ
Fallar si el gestor de colas se está desactivando temporalmente.

Esta opción fuerza que la llamada MQPUT o MQPUT1 falle si el gestor de colas está en estado de desactivación temporal.

La llamada devuelve el código de terminación CCFAIL con el código de razón RC2161 .

Opción predeterminada: si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

PMNONE
No se ha especificado ninguna opción.

Este valor puede utilizarse para indicar que no se ha especificado ninguna otra opción; todas las opciones asumen sus valores predeterminados. PMNONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *PMOPT* es PMNONE.

PMPRF (entero con signo de 10 dígitos)

Distintivos que indican qué campos MQPMR están presentes.

Este campo contiene distintivos que se deben establecer para indicar qué campos MQPMR están presentes en los registros de mensajes colocados proporcionados por la aplicación. *PMPRF* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *PMREC* es cero, o si *PMPRO* y *PMPRP* son cero.

Para los campos que están presentes, el gestor de colas utiliza para cada destino los valores de los campos del registro de mensajes de colocación correspondiente. Para los campos que están ausentes, el gestor de colas utiliza los valores de la estructura MQMD.

Se pueden especificar uno o varios de los distintivos siguientes para indicar qué campos están presentes en los registros de colocación de mensajes:

IDPFM
El campo de identificador de mensaje está presente.

PICD
El campo de identificador de correlación está presente.

IDPFG
El campo identificador de grupo está presente.

PFFB
El campo de comentarios está presente.

PFACC
El campo de señal de contabilidad está presente.

Si se especifica este distintivo, se debe especificar PMSETI o PMSETA en el campo *PMOPT* ; si no se cumple esta condición, la llamada falla con el código de razón RC2158 .

Si no hay campos MQPMR presentes, se puede especificar lo siguiente:

PNONE
No hay campos de registro de colocación de mensaje.

Si se especifica este valor, *PMREC* debe ser cero, o bien *PMPRO* y *PMPRP* deben ser cero.

PFNONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Si *PMPRF* contiene distintivos que no son válidos, o se proporcionan registros de colocación de mensajes pero *PMPRF* tiene el valor PFNONE, la llamada falla con el código de razón RC2158 .

Este es un campo de entrada. El valor inicial de este campo es PFNONE. Este campo se ignora si *PMVER* es menor que PMVER2.

PMPRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de mensaje de colocación desde el inicio de MQPMO.

Es el desplazamiento en bytes del primer registro de mensajes de colocación MQPMR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PMPRO* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *PMREC* es cero.

Cuando el mensaje se transfiere a una lista de distribución, se puede proporcionar una matriz de uno o más registros de mensajes de colocación MQPMR para especificar determinadas propiedades del mensaje para cada destino individualmente; estas propiedades son:

- Identificador de mensaje
- identificador de correlación
- Identificador de grupo
- valor de comentarios
- Señal de contabilidad

No es necesario especificar todas estas propiedades, pero independientemente del subconjunto que se elija, los campos deben especificarse en el orden correcto. Consulte la descripción de la estructura MQPMR para obtener más detalles.

Normalmente, debe haber tantos registros de mensajes de colocación como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cada registro de mensajes de colocación proporciona las propiedades de mensaje para la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben tener asignados registros de mensajes en las posiciones adecuadas de la matriz, aunque las propiedades del mensaje se ignoran en este caso.

Es posible que el número de registros de mensajes de colocación difiera del número de registros de objeto. Si hay menos registros de mensaje de colocación que registros de objeto, las propiedades de mensaje para los destinos que no tienen registros de mensaje de colocación se toman de los campos correspondientes en el MQMD del descriptor de mensaje. Si hay más registros de mensajes de colocación que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible

acceder a ellos). Los registros de mensajes de colocación son opcionales, pero si se proporcionan, debe haber *PMREC* de ellos.

Los registros de mensajes de colocación se pueden proporcionar de forma similar a los registros de objeto en MQOD, ya sea especificando un desplazamiento en *PMPRO*o especificando una dirección en *PMPRP* ; para obtener detalles sobre cómo hacerlo, consulte el campo *ODORO* descrito en "MQOD (Descriptor de objeto) en IBM i" en la página 1186.

No se puede utilizar más de uno de *PMPRO* y *PMPRP* ; la llamada falla con el código de razón RC2159 si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que PMVER2.

PMPRP (puntero)

Dirección del primer registro de mensaje de colocación.

Es la dirección del primer registro de mensaje de colocación MQPMR. *PMPRP* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *PMREC* es cero.

Se puede utilizar *PMPRP* o *PMPRO* para especificar los registros de mensajes de colocación, pero no ambos; consulte la descripción del campo PMRRO para obtener más detalles. Si no se utiliza *PMPRP* , debe establecerse en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *PMVER* es menor que PMVER2.

PMREC (entero con signo de 10 dígitos)

Número de registros de mensajes de colocación o registros de respuesta presentes.

Es el número de registros de mensajes de colocación MQPMR o registros de respuesta MQRR que ha proporcionado la aplicación. Este número puede ser mayor que cero sólo si el mensaje se está colocando en una lista de distribución. Los registros de mensajes de colocación y los registros de respuesta son opcionales; la aplicación no necesita proporcionar ningún registro, o puede optar por proporcionar registros de un solo tipo. Sin embargo, si la aplicación proporciona registros de ambos tipos, debe proporcionar registros *PMREC* de cada tipo.

No es necesario que el valor de *PMREC* sea el mismo que el número de destinos de la lista de distribución. Si se proporcionan demasiados registros, el exceso no se utiliza; si se proporcionan muy pocos registros, se utilizan los valores predeterminados para las propiedades de mensaje para los destinos que no tienen registros de mensajes de colocación (consulte *PMPRO* más adelante en este tema).

Si *PMREC* es menor que cero, o es mayor que cero pero el mensaje no se está colocando en una lista de distribución, la llamada falla con el código de razón RC2154 .

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que PMVER2.

PMRMN (serie de caracteres de 48 bytes)

Nombre resuelto del gestor de colas de destino.

Es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *PMRQN*, y puede ser el nombre del gestor de colas local.

Si *PMRQN* es una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *PMRMN* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *PMRQN* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona LNQMN. El valor inicial de este campo es de 48 caracteres en blanco.

PMRQN (serie de caracteres de 48 bytes)

Nombre resuelto de la cola de destino.

Es el nombre de la cola de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *PMRMN*.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

PMRRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de respuesta desde el inicio de MQPMO.

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PMRRO* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *PMREC* es cero.

Cuando el mensaje se transfiere a una lista de distribución, se puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas a las que no se ha enviado correctamente el mensaje (campo*RRCC* en MQRR) y el motivo de cada anomalía (campo*RRREA* en MQRR). Es posible que el mensaje no se haya enviado porque la cola no se ha podido abrir o porque la operación de colocación ha fallado. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunos mensajes se han enviado correctamente mientras que otros han fallado, o todos han fallado pero por motivos diferentes); el código de razón RC2136 de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro **REASON** de la llamada MQPUT o MQPUT1 y los registros de respuesta no se establecen.

Normalmente, debe haber tantos registros de respuesta como hay registros de objeto especificados por MQOD cuando se abre la lista de distribución; cuando es necesario, cada registro de respuesta se establece en el código de terminación y el código de razón para la colocación en la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir todavía deben tener registros de respuesta asignados en las posiciones adecuadas de la matriz, aunque se establecen en el código de terminación y el código de razón resultantes de la operación de apertura, en lugar de en la operación de colocación.

Es posible que el número de registros de respuesta difiera del número de registros de objeto. Si hay menos registros de respuesta que registros de objeto, es posible que la aplicación no pueda identificar todos los destinos para los que ha fallado la operación de colocación o las razones de las anomalías. Si hay más registros de respuesta que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible acceder a ellos). Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *PMREC* de ellos.

Los registros de respuesta se pueden proporcionar de forma similar a los registros de objeto en MQOD, especificando un desplazamiento en *PMRRO*o especificando una dirección en *PMRRP* ; para obtener detalles sobre cómo hacerlo, consulte el campo *ODORO* descrito en “MQOD (Descriptor de objeto) en IBM i” en la página 1186. Sin embargo, no se puede utilizar más de uno de *PMRRO* y *PMRRP* ; la llamada falla con el código de razón RC2156 si ambos son distintos de cero.

Para la llamada MQPUT1 , este campo debe ser cero. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que PMVER2.

PMRRP (puntero)

Dirección del primer registro de respuesta.

Es la dirección del primer registro de respuesta MQRR. *PMRRP* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *PMREC* es cero.

Se puede utilizar *PMRRP* o *PMRRO* para especificar los registros de respuesta, pero no ambos; consulte la descripción del campo PMRRO para obtener más detalles. Si no se utiliza *PMRRP* , debe establecerse en el puntero nulo o en bytes nulos.

Para la llamada MQPUT1 , este campo debe ser el puntero nulo o bytes nulos. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *PMVER* es menor que PMVER2.

PMSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

PMSIDV
Identificador de la estructura de opciones de colocación de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es PMSIDV.

PMSL (MQLONG)

El nivel de suscripción al que se ha dirigido esta publicación.

Sólo reciben esta publicación las suscripciones con el valor más alto *PMSL* menor o igual que este valor. Este valor debe estar en el rango de cero a 9; cero es el nivel más bajo.

El valor inicial de este campo es 9.

PMTO (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es -1.

PMUDC (entero con signo de 10 dígitos)

Número de mensajes enviados satisfactoriamente a colas remotas.

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que se resuelven en colas remotas. Los mensajes que el gestor de colas retiene temporalmente en formato de lista de distribución cuentan como el número de destinos individuales que contienen esas listas de distribución. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que PMVER2.

PMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

PMVER1
Estructura de opciones de colocación de Version-1 .

PMVER2
Estructura de opciones de colocación de Version-2 .

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

PMVERC
Versión actual de la estructura de opciones de colocación de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es PMVER1.

Valores iniciales

Tabla 719. Valores iniciales de los campos en MQPMO

Nombre de campo	Nombre de constante	Valor de constante
PMSID	PMSIDV	'PMO¬'
PMVER	PMVER1	1
PMOPT	PMNONE	0
PMTO	Ninguna	-1
PMCT	Ninguna	0
PMKDC	Ninguna	0
PMUDC	Ninguna	0
PMIDC	Ninguna	0
PMRQN	Ninguna	Espacios en blanco
PMRMN	Ninguna	Espacios en blanco
PMREC	Ninguna	0
PMPRF	PNONE	0
PMPRO	Ninguna	0
PMRRO	Ninguna	0
PMPRP	Ninguna	Puntero nulo o bytes nulos
PMRRP	Ninguna	Puntero nulo o bytes nulos

Nota:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Registro de colocación de mensajes) en IBM i

La estructura MQPMR se utiliza para especificar varias propiedades de mensaje para un único destino cuando se transfiere un mensaje a una lista de distribución.

Visión general

Finalidad: MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1 .

Conjunto de caracteres y codificación: Los datos de MQPMR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en la llamada MQPUT o MQPUT1 , es posible especificar valores diferentes para cada cola de destino en una lista de distribución. Algunos de los campos son sólo de entrada, otros son de entrada/salida.

Nota: Esta estructura es inusual en que no tiene un diseño fijo. Los campos de esta estructura son opcionales, y la presencia o ausencia de cada campo se indica mediante los distintivos del campo *PMPRF* en MQPMO. Los campos que están presentes ***deben aparecer en el siguiente orden*** :

- *PRMID*
- *PRCID*
- *PRGID*
- *PRFB*
- *PRACC*

Los campos ausentes no ocupan ningún espacio en el registro.

Puesto que MQPMR no tiene un diseño fijo, no se proporciona ninguna definición del mismo en el archivo COPY. El programador de aplicaciones debe crear una declaración que contenga los campos necesarios para la aplicación y establecer los distintivos en *PMPRF* para indicar los campos que están presentes.

- “Campos” en la página 1217
- “Valores iniciales” en la página 1218
- “Declaración RPG” en la página 1218

Campos

La estructura MQPMR contiene los campos siguientes; los campos se describen en **orden alfabético**:

PRACC (serie de bits de 32 bytes)

Señal de contabilidad.

Es la señal de contabilidad que se debe utilizar para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDACC* en MQMD para una colocación en una sola cola. Consulte la descripción de *MDACC* en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener información sobre el contenido de este campo.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

PRCID (serie de bits de 24 bytes)

Identificador de correlación.

Es el identificador de correlación que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDCID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRCID* .

Si se especifica PMNCID, se genera un *único* identificador de correlación nuevo y se utiliza para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa PMNMID (consulte el campo *PRMID*).

Es un campo de entrada/salida.

PRFB (entero con signo de 10 dígitos)

Código de retorno o de razón.

Este es el código de retorno que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDFB* en MQMD para una colocación en una sola cola.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

PRGID (serie de bits de 24 bytes)

Identificador de grupo.

Es el identificador de grupo que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDGID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRGID* . El valor se procesa tal como se documenta en Tabla 716 en la página 1205, pero con las diferencias siguientes:

- En los casos en los que se utilizaría un nuevo identificador de grupo, el gestor de colas genera un identificador de grupo diferente para cada destino (es decir, ningún destino tiene el mismo identificador de grupo).

- En los casos en los que se utilizaría el valor del campo, la llamada falla con el código de razón RC2258.

Es un campo de entrada/salida.

PRMID (serie de bits de 24 bytes)

Identificador del mensaje.

Es el identificador de mensaje que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDMID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRMID* . Si ese valor es MINONE, se genera un nuevo identificador de mensaje para *cada* de esos destinos (es decir, dos de esos destinos no tienen el mismo identificador de mensaje).

Si se especifica PMNMID, se generan nuevos identificadores de mensaje para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa PMNCID (consulte el campo *PRCID*).

Es un campo de entrada/salida.

Valores iniciales

No hay valores iniciales definidos para esta estructura, ya que no se proporciona ninguna declaración de estructura. La siguiente declaración de ejemplo muestra cómo el programador de aplicaciones debe declarar la estructura si se necesitan todos los campos.

Declaración 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 (Reglas y cabecera de formato) en IBM i

La estructura MQRFH define el diseño de las reglas y la cabecera de formato.

Visión general

Finalidad: esta cabecera se puede utilizar para enviar datos de serie en forma de pares nombre-valor.

Nombre de formato: FMRFH.

Conjunto de caracteres y codificación: los campos de la estructura MQRFH (incluido *RFNVS*) están en el juego de caracteres y la codificación proporcionados por los campos *MDCSI* y *MDENC* en la estructura de cabecera que precede a la MQRFH, o por los campos de la estructura MQMD si la MQRFH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

- “Campos” en la página 1219
- “Valores iniciales” en la página 1221
- “Declaración RPG” en la página 1221

Campos

La estructura MQRFH contiene los campos siguientes; los campos se describen en **orden alfabético**:

RFCSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen a *RFNVS*.

Especifica el identificador de juego de caracteres de los datos que siguen a *RFNVS* ; no se aplica a los datos de tipo carácter de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

Codificación numérica de datos que sigue a *RFNVS*.

Especifica la codificación numérica de los datos que siguen a *RFNVS* ; no se aplica a los datos numéricos de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RFFLG (entero con signo de 10 dígitos)

Distintivos.

Se puede especificar lo siguiente:

RFNONE
Sin distintivos.

El valor inicial de este campo es RFNONE.

RFFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen a *RFNVS*.

Especifica el nombre de formato de los datos que siguen a *RFNVS*.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

El valor inicial de este campo es FMNONE.

RFLEN (entero con signo de 10 dígitos)

Longitud total de MQRFH incluyendo *RFNVS*.

Es la longitud en bytes de la estructura MQRFH, incluido el campo *RFNVS* al final de la estructura. La longitud no incluye los datos de usuario que siguen al campo *RFNVS* .

Para evitar problemas con la conversión de datos de los datos de usuario en algunos entornos, considere la posibilidad de utilizar *RFLEN* como un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo el campo *RFNVS* :

RFLENV
Longitud de la parte fija de la estructura MQRFH.

El valor inicial de este campo es RFLENV.

RFNVS (serie de caracteres de n bytes)

Serie que contiene pares de nombre-valor.

Se trata de una serie de caracteres de longitud variable que contiene pares nombre-valor con el formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Cada nombre o valor debe estar separado del nombre o valor adyacente por uno o más caracteres en blanco; estos espacios en blanco no son significativos. Un nombre o valor puede contener espacios en blanco significativos añadiendo como prefijo y sufijo el nombre o valor con el carácter de comillas; todos los caracteres entre las comillas de apertura y las comillas de cierre coincidentes se tratan como significativos. En el ejemplo siguiente, el nombre es `FAMOUS_WORDS`y el valor es `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Un nombre o valor puede contener cualquier carácter que no sea el carácter nulo (que actúa como delimitador para *RFNVS*). Sin embargo, para ayudar a la interoperatividad, es posible que una aplicación prefiera restringir los nombres a los caracteres siguientes:

- Primer carácter: alfabético en mayúsculas o minúsculas (de la A a la Z, o de la a a la z), o subrayado.
- Caracteres subsiguientes: alfabético en mayúsculas o minúsculas, dígito decimal (de 0 a 9), subrayado, guión o punto.

Si un nombre o valor contiene una o más comillas, el nombre o valor debe especificarse entre comillas y cada comilla dentro de la serie debe duplicarse:

```
Famous_Words "The program displayed ""Hello World"""
```

Los nombres y valores distinguen entre mayúsculas y minúsculas, es decir, las letras minúsculas no se consideran iguales que las letras mayúsculas. Por ejemplo, `FAMOUS_WORDS` y `Famous_Words` son dos nombres diferentes.

La longitud en bytes de *RFNVS* es igual a *RFLEN* menos RFLENV. Para evitar problemas con la conversión de datos de los datos de usuario en algunos entornos, se recomienda que esta longitud sea un múltiplo de cuatro. *RFNVS* debe rellenarse con espacios en blanco hasta esta longitud, o terminarse antes colocando un carácter nulo después del último carácter significativo de la serie. El carácter nulo y los bytes que le siguen, hasta la longitud especificada de *RFNVS*, se ignoran.

Nota: Debido a que la longitud de este campo no es fija, el campo se omite de las declaraciones de la estructura que se proporcionan para los lenguajes de programación soportados.

RFSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RFSIDV
Identificador de reglas y estructura de cabecera de formato.

El valor inicial de este campo es RFSIDV.

RFVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RFVER1
Reglas de Version-1 y estructura de cabecera de formato.

El valor inicial de este campo es RFVER1.

Valores iniciales

Tabla 720. Valores iniciales de los campos en MQRFH

Nombre de campo	Nombre de constante	Valor de constante
RFSID	RFSIDV	'RFH¬'
RFVER	RFVER1	1
RFLEN	RFLENV	32
RFENC	ENNAT	Depende del entorno
RFCSI	CUNDF	0
RFFMT	FMNONE	Espacios en blanco
RFFLG	RFNONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (cabecera de reglas y formateo 2) en IBM i

La estructura MQRFH2 define el formato de las reglas version-2 y la cabecera de formato.

Visión general

Finalidad: Esta cabecera se puede utilizar para enviar datos que se han codificado utilizando una sintaxis similar a XML. Un mensaje puede contener dos o más estructuras MQRFH2 en serie, con datos de usuario opcionalmente a continuación de la última estructura MQRFH2 de la serie.

Nombre de formato: FMRFH2.

Conjunto de caracteres y codificación: las reglas especiales se aplican al juego de caracteres y a la codificación utilizados para la estructura MQRFH2 :

- Los campos que no son *RF2NVD* están en el juego de caracteres y la codificación proporcionados por los campos *MDCSI* y *MDENC* en la estructura de cabecera que precede a MQRFH2, o por esos campos en la estructura MQMD si MQRFH2 está al principio de los datos del mensaje de aplicación.

  El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

  Cuando se especifica GMCONV en la llamada MQGET, el gestor de colas convierte estos campos en el juego de caracteres y la codificación solicitados.
- *RF2NVD* está en el juego de caracteres proporcionado por el campo *RF2NVC* . Sólo determinados juegos de caracteres Unicode son válidos para *RF2NVC* (consulte la descripción de *RF2NVC* para obtener más detalles).

  Algunos juegos de caracteres tienen una representación que depende de la codificación. Si *RF2NVC* es uno de estos juegos de caracteres, *RF2NVD* debe estar en la misma codificación que los otros campos de la MQRFH2.

  Cuando se especifica GMCONV en la llamada MQGET, el gestor de colas convierte *RF2NVD* a la codificación solicitada, pero no cambia su juego de caracteres.

- "Campos" en la página 1222
- "Valores iniciales" en la página 1227
- "Declaración RPG" en la página 1227

Campos

La estructura MQRFH2 contiene los campos siguientes; los campos se describen en **orden alfabético**:

RF2CSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen al último campo *RF2NVD* .

Especifica el identificador de juego de caracteres de los datos que siguen al último campo *RF2NVD* ; no se aplica a los datos de tipo carácter de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSINHT.

RF2ENC (entero con signo de 10 dígitos)

Codificación numérica de los datos que siguen al último campo *RF2NVD* .

Especifica la codificación numérica de los datos que siguen al último campo *RF2NVD* ; no se aplica a los datos numéricos de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RF2FLG (entero con signo de 10 dígitos)

Distintivos.

Se debe especificar el valor siguiente:

RFNONE
Sin distintivos.

El valor inicial de este campo es RFNONE.

RF2FMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen al último campo *RF2NVD* .

Especifica el nombre de formato de los datos que siguen al último campo *RF2NVD* .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

El valor inicial de este campo es FMNONE.

RF2LEN (entero con signo de 10 dígitos)

Longitud total de MQRFH2 incluidos todos los campos *RF2NVL* y *RF2NVD* .

Es la longitud en bytes de la estructura MQRFH2 , incluidos los campos *RF2NVL* y *RF2NVD* al final de la estructura. Es válido para que haya varios pares de campos *RF2NVL* y *RF2NVD* al final de la estructura, en la secuencia:

```
length1, data1, length2, data2, ...
```

RF2LEN no incluye ningún dato de usuario que pueda seguir al último campo de *RF2NVD* al final de la estructura.

Para evitar problemas con la conversión de datos de los datos de usuario en algunos entornos, considere la posibilidad de utilizar *RF2LEN* como un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo los campos *RF2NVL* y *RF2NVD* :

RFLEN2
Longitud de la parte fija de la estructura MQRFH2 .

El valor inicial de este campo es RFLEN2.

RF2NVC (entero con signo de 10 dígitos)

Identificador de juego de caracteres de *RF2NVD*.

Especifica el identificador de juego de caracteres codificado de los datos en el campo *RF2NVD* . Es diferente del juego de caracteres de las otras series de la estructura MQRFH2 y puede ser diferente del juego de caracteres de los datos (si los hay) que siguen al último campo *RF2NVD* al final de la estructura.

RF2NVC debe tener uno de los siguientes valores de CCSID:

1200
UTF-16, versión soportada más reciente de Unicode

13488
UTF-16, subconjunto de Unicode versión 2.0

17584
UTF-16, subconjunto de Unicode versión 3.0 (incluye símbolo del Euro)

1208
UTF-8, versión soportada más reciente de Unicode

Para los juegos de caracteres UTF-16 , la codificación (orden de bytes) de *RF2NVD* debe ser la misma que la codificación de los otros campos de la estructura MQRFH2 . Los caracteres sustitutos (X'D800'a X'DFFF') no están soportados.

Nota: Si *RF2NVC* no tiene uno de los valores listados anteriormente y la estructura MQRFH2 requiere conversión en la llamada MQGET, la llamada se completa con el código de razón RC2111 y el mensaje se devuelve sin convertir.

El valor inicial de este campo es 1208.

RF2NVD (serie de caracteres de n bytes)

Datos de nombre/valor.

Se trata de una serie de caracteres de longitud variable que contiene datos codificados utilizando una sintaxis similar a XML. La longitud en bytes de esta serie la proporciona el campo *RF2NVL* que precede al campo *RF2NVD* ; esta longitud debe ser un múltiplo de cuatro.

Los campos *RF2NVL* y *RF2NVD* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

RF2NVD no es habitual porque no se convierte al juego de caracteres especificado en la llamada MQGET cuando el mensaje se recupera con la opción GMCONV en vigor; *RF2NVD* permanece en su juego de caracteres original. Sin embargo, *RF2NVD* se convierte a la codificación especificada en la llamada MQGET.

Sintaxis de datos de nombre/valor: la serie consta de una única "carpeta" que contiene cero o más propiedades. La carpeta está delimitada por etiquetas de inicio y finalización XML con el mismo nombre que la carpeta:

```
<folder> property1 property2 ... </folder>
```

Los caracteres que siguen a la etiqueta de fin de carpeta, hasta la longitud definida por *RF2NVL*, deben estar en blanco. Dentro de la carpeta, cada propiedad se compone de un nombre y un valor, y opcionalmente un tipo de datos:

```
<name dt="datatype">value</name>
```

En estos ejemplos:

- Los caracteres delimitadores (<, =, ",/y >) deben especificarse exactamente como se muestra.
- `name` es el nombre especificado por el usuario de la propiedad; consulte el ejemplo siguiente para obtener más información sobre los nombres.
- `datatype` es un tipo de datos opcional especificado por el usuario de la propiedad; consulte el ejemplo siguiente para ver los tipos de datos válidos.
- `value` es el valor especificado por el usuario de la propiedad; consulte los párrafos siguientes para obtener más información sobre los valores.
- Los espacios en blanco son significativos entre el carácter > que precede a un valor y el carácter < que sigue al valor, y al menos un espacio en blanco debe preceder a `dt=`. En otros lugares, los espacios en blanco se pueden codificar libremente entre etiquetas, o antes o después de etiquetas (por ejemplo, para mejorar la legibilidad); estos espacios en blanco no son significativos.

Si las propiedades están relacionadas entre sí, se pueden agrupar encerrándolas dentro de etiquetas de inicio y finalización XML con el mismo nombre que el grupo:

```
<folder> <group> property1 property2 ... </group> </folder>
```

Los grupos se pueden anidar dentro de otros grupos, sin límite, y un grupo puede aparecer más de una vez dentro de una carpeta. También es válido que una carpeta contenga algunas propiedades en grupos y otras propiedades no en grupos.

Nombres de propiedades, grupos y carpetas: los nombres de propiedades, grupos y carpetas deben ser nombres de etiqueta XML válidos, con la excepción del carácter de dos puntos, que no está permitido en un nombre de propiedad, grupo o carpeta. En concreto:

- Los nombres deben empezar por una letra o un subrayado. Las letras válidas se definen en la especificación XML W3C y constan esencialmente de las categorías Unicode Ll, Lu, Lo, Lt y Nl.
- Los caracteres restantes de un nombre pueden ser letras, dígitos decimales, subrayados, guiones o puntos. Estos corresponden a las categorías Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm y Nd.
- Los caracteres de compatibilidad Unicode (X'F900' y superiores) no están permitidos en ninguna parte de un nombre.
- Los nombres no deben empezar por la serie `XML` en ninguna mezcla de mayúsculas o minúsculas.

Además:

- Los nombres distinguen entre mayúsculas y minúsculas. Por ejemplo, `ABC`, `abc`y `Abc` son tres nombres diferentes.
- Cada carpeta tiene un espacio de nombres independiente. Como resultado, un grupo o propiedad de una carpeta no entra en conflicto con un grupo o propiedad del mismo nombre en otra carpeta.
- Los grupos y las propiedades ocupan el mismo espacio de nombres dentro de una carpeta. Como resultado, una propiedad no puede tener el mismo nombre que un grupo dentro de la carpeta que contiene dicha propiedad.

Generalmente, los programas que analizan el campo *RF2NVD* deben ignorar las propiedades o grupos que tienen nombres que el programa no reconoce, siempre que esas propiedades o grupos estén formados correctamente.

Tipos de datos de propiedades: cada propiedad puede tener un tipo de datos opcional. Si se especifica, el tipo de datos debe ser uno de los siguientes valores, en mayúsculas, minúsculas o mayúsculas y minúsculas:

Tabla 721. Tipos de datos y su uso

Tipo de datos	Se utiliza para
`string`	Cualquier secuencia de caracteres. Se deben especificar determinados caracteres utilizando secuencias de escape.
`boolean`	El carácter 0 o 1 (1 indica TRUE).
`bin.hex`	Dígitos hexadecimales que representan octetos.
`i1`	Número entero en el rango de -128 a +127, expresado utilizando sólo dígitos decimales y signo opcional.
`i2`	Número entero en el rango de -32 768 a +32 767, expresado utilizando sólo dígitos decimales y signo opcional.
`i4`	Número entero en el rango de -2 147 483 648 a + 2 147 483 647, expresado utilizando sólo dígitos decimales y signo opcional.
`i8`	Número entero en el rango de -9 223 372 036 854 775 808 a + 9 223 372 036 854 775 807, expresado utilizando sólo dígitos decimales y signo opcional.

Tabla 721. Tipos de datos y su uso (continuación)	
Tipo de datos	**Se utiliza para**
`int`	Número entero en el rango de -9 223 372 036 854 775 808 a + 9 223 372 036 854 775 807, expresado utilizando sólo dígitos decimales y signo opcional. Esto se puede utilizar en lugar de `i1`, `i2`, `i4`o `i8` si el remitente no desea que implique una precisión determinada.
`r4`	Número de coma flotante con una magnitud en el rango de 1.175E-37 a 3.402 823 47E+38, expresado utilizando dígitos decimales, signo opcional, dígitos fraccionarios opcionales y exponente opcional.
`r8`	Número de coma flotante con magnitud en el rango de 2.225E-307 a 1.797 693 134 862 3E+308 expresado utilizando dígitos decimales, signo opcional, dígitos fraccionarios opcionales y exponente opcional.

Valores de las propiedades: El valor de una propiedad puede constar de cualquier carácter, excepto tal como se detalla en la tabla siguiente. Cada aparición en el valor de un carácter marcado como "obligatorio" debe sustituirse por la secuencia de escape correspondiente. Cada aparición en el valor de un carácter marcado como "opcional" se puede sustituir por la secuencia de escape correspondiente, pero esto no es necesario.

Tabla 722. Caracteres de escape y su uso		
Carácter	**Secuencia de escape**	**Utilización**
`&`	`&`	Obligatorio
`<`	`<`	Obligatorio
`>`	`>`	Opcional
`"`	`"`	Opcional
`'`	`'`	Opcional

Nota: El carácter & al principio de una secuencia de escape no debe sustituirse por `&`.

En el ejemplo siguiente, los espacios en blanco del valor son significativos; sin embargo, no se necesitan secuencias de escape:

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (entero con signo de 10 dígitos)

Longitud de *RF2NVD*.

Especifica la longitud en bytes de los datos del campo *RF2NVD* . Para evitar problemas con la conversión de datos de los datos (si los hay) que *sigue* al campo *RF2NVD* , *RF2NVL* debe ser un múltiplo de cuatro.

Nota: Los campos *RF2NVL* y *RF2NVD* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

RF2SID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RFSIDV
Identificador de reglas y estructura de cabecera de formato.

El valor inicial de este campo es RFSIDV.

RF2VER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RFVER2
Version-2 reglas y estructura de cabecera de formato.

El valor inicial de este campo es RFVER2.

Valores iniciales

Tabla 723. Valores iniciales de los campos en MQRFH2

Nombre de campo	Nombre de constante	Valor de constante
RF2SID	RFSIDV	'RFH¬'
RF2VER	RFVER2	2
RF2LEN	RFLEN2	36
RF2ENC	ENNAT	Depende del entorno
RF2CSI	CSINHT	-2
RF2FMT	FMNONE	Espacios en blanco
RF2FLG	RFNONE	0
RF2NVC	Ninguna	1208

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (cabecera de mensaje de referencia) en IBM i

La estructura MQRMH define el formato de una cabecera de mensaje de referencia.

Visión general

Finalidad: esta cabecera se utiliza con salidas de canal de mensajes escritas por el usuario para enviar grandes cantidades de datos (denominados "datos masivos") de un gestor de colas a otro. La diferencia en comparación con la mensajería normal es que los datos masivos no se almacenan en una cola; en su lugar, sólo se almacena en la cola una *referencia* a los datos masivos. Esto reduce la posibilidad de que unos pocos mensajes grandes agoten los recursos de IBM MQ .

Nombre de formato: FMRMH.

Conjunto de caracteres y codificación: los datos de caracteres en MQRMH, y las series a las que se dirigen los campos de desplazamiento, deben estar en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **`CodedCharSetId`** . Los datos numéricos en MQRMH deben estar en la codificación de máquina nativa; esto viene dado por el valor de ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación de MQRMH deben establecerse en los campos *`MDCSI`* y *`MDENC`* en:

- El MQMD (si la estructura MQRMH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQRMH (todos los demás casos).

Uso: una aplicación coloca un mensaje que consta de una MQRMH, pero omitiendo los datos masivos. Cuando un agente de canal de mensajes (MCA) lee el mensaje de la cola de transmisión, se invoca una salida de mensaje proporcionada por el usuario para procesar la cabecera de mensaje de referencia. La salida puede añadir al mensaje de referencia los datos masivos identificados por la estructura MQRMH, antes de que el MCA envíe el mensaje a través del canal al siguiente gestor de colas.

En el extremo receptor, debe existir una salida de mensajes que espere mensajes de referencia. Cuando se recibe un mensaje de referencia, la salida debe crear el objeto a partir de los datos masivos que siguen a la MQRMH en el mensaje y, a continuación, pasar el mensaje de referencia sin los datos masivos. El mensaje de referencia puede ser recuperado posteriormente por una aplicación que lee el mensaje de referencia (sin los datos masivos) de una cola.

Normalmente, la estructura MQRMH es todo lo que está en el mensaje. Sin embargo, si el mensaje está en una cola de transmisión, una o más cabeceras adicionales precederán a la estructura MQRMH.

También se puede enviar un mensaje de referencia a una lista de distribución. En este caso, la estructura MQDH y sus registros relacionados preceden a la estructura MQRMH cuando el mensaje está en una cola de transmisión.

Nota: Un mensaje de referencia no debe enviarse como un mensaje segmentado, porque la salida de mensaje no puede procesarlo correctamente.

- “Conversión de datos” en la página 1228
- “Campos” en la página 1229
- “Valores iniciales” en la página 1233
- “Declaración RPG” en la página 1234

Conversión de datos

A efectos de conversión de datos, la conversión de la estructura MQRMH incluye la conversión de los datos del entorno de origen, el nombre del objeto de origen, los datos del entorno de destino y el nombre del objeto de destino. Cualquier otro byte dentro de *`RMLEN`* bytes del inicio de la estructura se descarta o tiene valores no definidos después de la conversión de datos. Los datos masivos se convertirán siempre que todas las sentencias siguientes sean verdaderas:

- Los datos masivos están presentes en el mensaje cuando se realiza la conversión de datos.

- El campo *RMFMT* en MQRMH tiene un valor distinto de FMNONE.
- Existe una salida de conversión de datos escrita por el usuario con el nombre de formato especificado.

No obstante, tenga en cuenta que normalmente los datos masivos no están presentes en el mensaje cuando el mensaje está en una cola y que, como resultado, la opción GMCONV no convertirá los datos masivos.

Campos

La estructura MQRMH contiene los campos siguientes; los campos se describen en **orden alfabético**:

RMCSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de datos masivos.

Especifica el identificador de juego de caracteres de los datos masivos; no se aplica a los datos de tipo carácter de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

RMDEL (entero con signo de 10 dígitos)

Longitud de los datos de entorno de destino.

Si este campo es cero, no hay datos de entorno de destino y se ignora *RMDEO* .

RMDEO (entero con signo de 10 dígitos)

Desplazamiento de los datos de entorno de destino.

Este campo especifica el desplazamiento de los datos de entorno de destino desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos del entorno de destino, si el creador los conoce. Por ejemplo, los datos de entorno de destino pueden ser la vía de acceso del directorio del objeto donde se van a almacenar los datos masivos. Sin embargo, si el creador no conoce los datos de entorno de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar la información de entorno necesaria.

La longitud de los datos de entorno de destino la proporciona *RMDEL* ; si esta longitud es cero, no hay datos de entorno de destino y se ignora *RMDEO* . Si está presente, los datos de entorno de destino deben residir completamente dentro de *RMLEN* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos del entorno de destino son contiguos a ninguno de los datos direccionado por los campos *RMSEO*, *RMSNO*y *RMDNO* .

El valor inicial de este campo es 0.

RMDL (entero con signo de 10 dígitos)

Longitud de datos masivos.

El campo *RMDL* especifica la longitud de los datos masivos a los que hace referencia la estructura MQRMH.

Si los datos masivos están presentes en el mensaje, los datos empiezan en un desplazamiento de *RMLEN* bytes desde el inicio de la estructura MQRMH. La longitud de todo el mensaje menos *RMLEN* proporciona la longitud de los datos masivos presentes.

Si los datos están presentes en el mensaje, *RMDL* especifica la cantidad de datos que son relevantes. El caso normal es que *RMDL* tenga el mismo valor que la longitud de los datos presentes en el mensaje.

Si la estructura MQRMH representa los datos restantes en el objeto (empezando por el desplazamiento lógico especificado), se puede utilizar el valor cero para *RMDL*, si los datos masivos no están presentes en el mensaje.

Si no hay datos presentes, el final de MQRMH coincide con el final del mensaje.

El valor inicial de este campo es 0.

RMDNL (entero con signo de 10 dígitos)

Longitud del nombre de objeto de destino.

Si este campo es cero, no hay ningún nombre de objeto de destino y se ignora *RMDNO* .

RMDNO (entero con signo de 10 dígitos)

Desplazamiento del nombre de objeto de destino.

Este campo especifica el desplazamiento del nombre de objeto de destino desde el inicio de la estructura MQRMH. El nombre de objeto de destino puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario identificar el objeto que se va a crear o modificar.

La longitud del nombre de objeto de destino viene dada por *RMDNL* ; Si esta longitud es cero, no hay ningún nombre de objeto de destino y se ignora *RMDNO* . Si está presente, el nombre de objeto de destino debe residir completamente dentro de *RMLEN* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de destino es contiguo a cualquiera de los datos a los que se dirigen los campos *RMSEO*, *RMSNO*y *RMDEO* .

El valor inicial de este campo es 0.

RMDO (entero con signo de 10 dígitos)

Desplazamiento bajo de datos masivos.

Este campo especifica el desplazamiento bajo de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. El desplazamiento de los datos masivos desde el inicio del objeto se denomina *desplazamiento lógico*. Este no es el desplazamiento físico de los datos masivos desde el inicio de la estructura MQRMH-ese desplazamiento lo proporciona *RMLEN*.

Para permitir que se envíen objetos grandes utilizando mensajes de referencia, el desplazamiento lógico se divide en dos campos, y el desplazamiento lógico real viene dado por la suma de estos dos campos:

- *RMDO* representa el resto obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es un valor comprendido entre 0 y 999.999.999.
- *RMDO2* representa el resultado obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es el número de múltiplos completos de 1 000 000 000 000 que existen en el desplazamiento lógico. El número de múltiplos está en el rango de 0 a 999.999.999.

El valor inicial de este campo es 0.

RMDO2 (entero con signo de 10 dígitos)

Desplazamiento alto de datos masivos.

Este campo especifica el desplazamiento alto de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. Es un valor comprendido entre 0 y 999.999.999. Consulte *RMDO* para obtener detalles.

El valor inicial de este campo es 0.

RMENC (entero con signo de 10 dígitos)

Codificación numérica de datos masivos.

Especifica la codificación numérica de los datos masivos; no se aplica a los datos numéricos de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RMFLG (entero con signo de 10 dígitos)

Distintivos de mensaje de referencia.

Se definen los distintivos siguientes:

RMLAST
El mensaje de referencia contiene o representa la última parte del objeto.

Este distintivo indica que el mensaje de referencia representa o contiene la última parte del objeto referenciado.

RMNLST
El mensaje de referencia no contiene ni representa la última parte del objeto.

RMNLST está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial de este campo es RMNLST.

RMFMT (serie de caracteres de 8 bytes)

Nombre de formato de datos masivos.

Especifica el nombre de formato de los datos masivos.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

El valor inicial de este campo es FMNONE.

RMLEN (entero con signo de 10 dígitos)

Longitud total de MQRMH, incluidas las series al final de los campos fijos, pero no los datos masivos.

El valor inicial de este campo es cero.

RMOII (serie de bits de 24 bytes)

Identificador de instancia de objeto.

Este campo se puede utilizar para identificar una instancia específica de un objeto. Si no es necesario, debe establecerse en el valor siguiente:

OIINON
No se ha especificado ningún identificador de instancia de objeto.

El valor es cero binario para la longitud del campo.

La longitud de este campo la proporciona LNOIID. El valor inicial de este campo es OIINON.

RMOT (serie de caracteres de 8 bytes)

Tipo de objeto.

Es un nombre que puede utilizar la salida de mensajes para reconocer los tipos de mensajes de referencia a los que da soporte. Considere la posibilidad de hacer que el nombre se ajuste a las mismas reglas que el campo *RMFMT* .

El valor inicial de este campo es de 8 blancos.

RMSEL (entero con signo de 10 dígitos)

Longitud de los datos de entorno de origen.

Si este campo es cero, no hay datos de entorno de origen y se ignora *RMSEO* .

El valor inicial de este campo es 0.

RMSEO (entero con signo de 10 dígitos)

Desplazamiento de datos de entorno de origen.

Este campo especifica el desplazamiento de los datos de entorno de origen desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos de entorno de origen, si el creador los conoce. Por ejemplo, los datos de entorno de origen pueden ser la vía de acceso del directorio del objeto que contiene los datos masivos. Sin embargo, si el creador no conoce los datos del entorno de origen, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar la información de entorno necesaria.

La longitud de los datos de entorno de origen la proporciona *RMSEL* ; si esta longitud es cero, no hay datos de entorno de origen y se ignora *RMSEO* . Si está presente, los datos de entorno de origen deben residir completamente dentro de *RMLEN* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos de entorno se inician inmediatamente después del último campo fijo de la estructura o que son contiguos con cualquiera de los datos direccionados por los campos *RMSNO*, *RMDEO*y *RMDNO* .

El valor inicial de este campo es 0.

RMSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RMSIDV
Identificador de la estructura de cabecera de mensaje de referencia.

El valor inicial de este campo es RMSIDV.

RMSNL (entero con signo de 10 dígitos)

Longitud del nombre de objeto de origen.

Si este campo es cero, no hay ningún nombre de objeto de origen y se ignora *RMSNO* .

El valor inicial de este campo es 0.

RMSNO (entero con signo de 10 dígitos)

Desplazamiento del nombre de objeto de origen.

Este campo especifica el desplazamiento del nombre de objeto de origen desde el inicio de la estructura MQRMH. El nombre de objeto de origen puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de origen, es responsabilidad de la salida de mensaje proporcionada por el usuario identificar el objeto al que se debe acceder.

La longitud del nombre de objeto de origen viene dada por *RMSNL* ; si esta longitud es cero, no hay ningún nombre de objeto de origen y se ignora *RMSNO* . Si está presente, el nombre de objeto de origen debe residir completamente dentro de *RMLEN* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de origen es contiguo a cualquiera de los datos a los que se dirigen los campos *RMSEO*, *RMDEO*y *RMDNO* .

El valor inicial de este campo es 0.

RMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RMVER1
Estructura de cabecera de mensaje de referencia Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

RMVERC
Versión actual de la estructura de cabecera de mensaje de referencia.

El valor inicial de este campo es RMVER1.

Valores iniciales

Tabla 724. Valores iniciales de los campos en MQRMH

Nombre de campo	Nombre de constante	Valor de constante
RMSID	RMSIDV	'RMH¬'
RMVER	RMVER1	1
RMLEN	Ninguna	0
RMENC	ENNAT	Depende del entorno
RMCSI	CUNDF	0
RMFMT	FMNONE	Espacios en blanco
RMFLG	RMNLST	0
RMOT	Ninguna	Espacios en blanco
RMOII	OIINON	Nulos
RMSEL	Ninguna	0
RMSEO	Ninguna	0
RMSNL	Ninguna	0
RMSNO	Ninguna	0
RMDEL	Ninguna	0
RMDEO	Ninguna	0
RMDNL	Ninguna	0
RMDNO	Ninguna	0
RMDL	Ninguna	0
RMDO	Ninguna	0
RMDO2	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
```

```
     D*
     D* MQRMH Structure
     D*
     D* Structure identifier
     D  RMSID                  1      4    INZ('RMH ')
     D* Structure version number
     D  RMVER                  5      8I 0 INZ(1)
     D* Total length of MQRMH, includingstrings at end of fixed fields,but not
     D* the bulk data
     D  RMLEN                  9     12I 0 INZ(0)
     D* Numeric encoding of bulk data
     D  RMENC                 13     16I 0 INZ(273)
     D* Character set identifier of bulkdata
     D  RMCSI                 17     20I 0 INZ(0)
     D* Format name of bulk data
     D  RMFMT                 21     28    INZ('        ')
     D* Reference message flags
     D  RMFLG                 29     32I 0 INZ(0)
     D* Object type
     D  RMOT                  33     40    INZ
     D* Object instance identifier
     D  RMOII                 41     64    INZ(X'00000000000000-
     D                                     00000000000000000000-
     D                                     000000000000')
     D* Length of source environmentdata
     D  RMSEL                 65     68I 0 INZ(0)
     D* Offset of source environmentdata
     D  RMSEO                 69     72I 0 INZ(0)
     D* Length of source object name
     D  RMSNL                 73     76I 0 INZ(0)
     D* Offset of source object name
     D  RMSNO                 77     80I 0 INZ(0)
     D* Length of destination environmentdata
     D  RMDEL                 81     84I 0 INZ(0)
     D* Offset of destination environmentdata
     D  RMDEO                 85     88I 0 INZ(0)
     D* Length of destination objectname
     D  RMDNL                 89     92I 0 INZ(0)
     D* Offset of destination objectname
     D  RMDNO                 93     96I 0 INZ(0)
     D* Length of bulk data
     D  RMDL                  97    100I 0 INZ(0)
     D* Low offset of bulk data
     D  RMDO                 101    104I 0 INZ(0)
     D* High offset of bulk data
     D  RMDO2                105    108I 0 INZ(0)
```

Declaración RPG

IBM i MQRR (registro de respuesta) en IBM i

La estructura MQRR se utiliza para recibir el código de terminación y el código de razón resultantes de la operación de apertura o colocación para una sola cola de destino, cuando el destino es una lista de distribución.

Visión general

Finalidad: MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1 .

Conjunto de caracteres y codificación: los datos de MQRR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en las llamadas MQOPEN y MQPUT, o en la llamada MQPUT1 , es posible determinar los códigos de terminación y los códigos de razón para todas las colas de una lista de distribución cuando el resultado de la llamada se mezcla, es decir, cuando la llamada es satisfactoria para algunas colas de la lista pero falla para otras. El código de razón RC2136 de la llamada indica que el gestor de colas ha establecido los registros de respuesta (si los proporciona la aplicación).

- “Campos” en la página 1235

- “Valores iniciales” en la página 1235
- “Declaración RPG” en la página 1235

Campos

La estructura MQRR contiene los campos siguientes; los campos se describen en **orden alfabético**:

RRCC (entero con signo de 10 dígitos)

Código de terminación para cola.

Es el código de terminación resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es CCOK.

RRREA (entero con signo de 10 dígitos)

Código de razón para la cola.

Este es el código de razón resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es RCNONE.

Valores iniciales

Tabla 725. Valores iniciales de los campos en MQRR

Nombre de campo	Nombre de constante	Valor de constante
RRCC	CCOK	0
RRREA	RCNONE	0

Declaración 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 (opciones de configuración de TLS) en IBM i

La estructura MQSCO (con los campos TLS en la estructura MQCD) permite a una aplicación que se ejecuta como IBM MQ MQI client especificar opciones de configuración que controlan el uso de TLS para la conexión de cliente cuando el protocolo de canal es TCP/IP.

Visión general

Finalidad: la estructura es un parámetro de entrada en la llamada MQCONNX.

Si el protocolo de canal para el canal de cliente no es TCP/IP, se ignora la estructura MQSCO.

Conjunto de caracteres y codificación: los datos de MQSCO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

- “Campos” en la página 1236
- “Valores iniciales” en la página 1239
- “Declaración RPG” en la página 1240

Campos

La estructura MQSCO contiene los campos siguientes; los campos se describen en **orden alfabético**:

SCAIC (entero con signo de 10 dígitos)

Es el número de registros de información de autenticación (MQAIR) a los que se dirigen los campos *SCAIP* o *SCAIO* . Para obtener más información, consulte“MQAIR (registro de información de autenticación) en IBM i” en la página 1035. El valor debe ser mayor o igual que cero. Si el valor no es válido, la llamada falla con el código de razón RC2383.

Este es un campo de entrada. El valor inicial de este campo es 0.

SCAIO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes del primer registro de información de autenticación desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo. El campo se ignora si *SCAIC* es cero.

Puede utilizar *SCAIO* o *SCAIP* para especificar los registros MQAIR, pero no ambos; consulte la descripción del campo *SCAIP* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

SCAIP (entero con signo de 10 dígitos)

Es la dirección del primer registro de información de autenticación. El campo se ignora si *SCAIC* es cero.

Puede proporcionar la matriz de registros MQAIR de una de estas dos maneras:

- Utilizando el campo de puntero *SCAIP*

  En este caso, la aplicación puede declarar una matriz de registros MQAIR separada de la estructura MQSCO y establecer *SCAIP* en la dirección de la matriz.

  Considere la posibilidad de utilizar *SCAIP* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

- Utilizando el campo de desplazamiento *SCAIO*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga una MQSCO seguida de la matriz de registros MQAIR y establecer *SCAIO* en el desplazamiento del primer registro de la matriz desde el inicio de la estructura MQSCO. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Considere la posibilidad de utilizar *SCAIO* para lenguajes de programación que no dan soporte al tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Sea cual sea la técnica que elija, solo se puede utilizar uno de *SCAIP* y *SCAIO* ; la llamada falla con el código de razón RC2384 si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

SCCERLBL (entero con signo de 10 dígitos)

Este campo proporciona detalles de la etiqueta de certificado que se está utilizando.

IBM MQ inicializa el valor del campo SCCERLBL como espacios en blanco. Especifique el valor necesario o acepte el valor predeterminado.

`ibmwebspheremq`*`user_id`* es un valor válido para este campo para todas las versiones del producto y para las versiones de MQSCO inferiores a 5.0 es el único valor válido. Por lo tanto, el valor de este campo se interpreta en tiempo de ejecución y se cambia si es necesario. Si especifica una versión de MQSCO inferior a 5.0, o acepta el valor predeterminado de blancos para el campo SCCERLBL, el sistema utiliza el valor `ibmwebspheremq`*`user_id`*.

Este es un campo de entrada.

SCCERTVPOL (entero con signo de 10 dígitos)

Este campo especifica qué tipo de política de validación de certificados se utiliza. El campo se puede establecer en uno de los valores siguientes:

MQ_CERT_VAL_POLICY_ANY
: Aplique cada una de las políticas de validación de certificados soportadas por la biblioteca de sockets seguros. Acepte la cadena de certificados si alguna de las políticas considera válida la cadena de certificados.

MQ_CERT_VAL_POLICY_RFC5280
: Aplique sólo la política de validación de certificados compatible con RFC5280 . Este valor proporciona una validación más estricta que el valor ANY, pero rechaza algunos certificados digitales más antiguos.

El valor inicial de este campo es MQ_CERT_VAL_POLICY_ANY

SCCH (entero con signo de 10 dígitos)

Este campo proporciona detalles de configuración para el hardware criptográfico conectado al sistema cliente.

Establezca el campo en una serie en el formato siguiente, o déjelo en blanco o nulo:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting>;
```

Para utilizar hardware criptográfico que se ajuste a la interfaz PKCS11 , por ejemplo, IBM 4960 o IBM 4963, especifique la vía de acceso del controlador PKCS11 , la etiqueta de señal PKCS11 y las series de contraseña de señal PKCS11 , cada una terminada con un punto y coma.

La vía de acceso del controlador PKCS #11 es una vía de acceso absoluta a la biblioteca compartida que proporciona soporte para la tarjeta PKCS #11 . El nombre del archivo de controlador PKCS #11 es el nombre de la biblioteca compartida. Un ejemplo del valor necesario para la vía de acceso PKCS #11 y el nombre de archivo es:

```
/usr/lib/pkcs11/PKCS11_API.so
```

La etiqueta de señal PKCS #11 debe estar totalmente en minúsculas. Si ha configurado el hardware con una etiqueta de señal en mayúsculas o en mayúsculas, vuelva a configurarlo con esta etiqueta en minúsculas.

Si no es necesaria ninguna configuración de hardware criptográfico, establezca el campo en blanco o nulo.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, o conduce a una anomalía cuando se utiliza para configurar el hardware de cifrado, la llamada falla con el código de razón RC2382.

Este es un campo de entrada. La longitud de este campo la proporciona LNSSCH. El valor inicial de este campo es de caracteres en blanco.

SCEPSUITEB (entero con signo de 10 dígitos)

Este campo especifica si se utiliza la criptografía compatible con Suite B y qué nivel de fuerza se emplea. El valor puede ser uno o varios de los siguientes:

- SCEPSUITEB0

  No se utiliza la criptografía compatible con Suite B.
- SCEPSUITEB1

  Se utiliza la seguridad de potencia de 128 bits de Suite B.
- SCEPSUITEB2

  Se utiliza la seguridad de potencia de 192 bits de la suite B.

Nota: El uso de SCEPSUITEB0 con cualquier otro valor en este campo no es válido.

SCFR (entero con signo de 10 dígitos)

IBM MQ se puede configurar con hardware criptográfico para que los módulos de criptografía utilizados sean los proporcionados por el producto de hardware; estos pueden estar certificados por FIPS a un nivel determinado en función del producto de hardware criptográfico en uso.

Utilice este campo para especificar que sólo se utilizan algoritmos certificados por FIPS si la criptografía se proporciona en el software proporcionado por IBM MQ.

Cuando se instala IBM MQ , también se instala una implementación de criptografía TLS que proporciona algunos módulos certificados por FIPS.

Los valores pueden ser:

MQSSL_FIPS_NO

Éste es el valor predeterminado. Cuando se establece en este valor:

- Se puede utilizar cualquier CipherSpec soportada en una plataforma determinada.
- Si se ejecuta sin utilizar hardware criptográfico, las CipherSpecs siguientes se ejecutan utilizando criptografía certificada FIPS 140-2 en las plataformas 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

Cuando se establece en este valor, a menos que esté utilizando hardware criptográfico para realizar la criptografía, puede estar seguro de que

- Solo se pueden utilizar algoritmos criptográficos con certificado FIPS en la CipherSpec que se aplica a esta conexión de cliente.
- Las conexiones de canal TLS de entrada y salida sólo son satisfactorias si se utiliza una de las siguientes especificaciones de cifrado:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

Notas:

1. La CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA está en desuso.
2. Siempre que sea posible, si se han configurado CipherSpecs sólo para FIPS, el cliente MQI rechaza las conexiones que especifican una CipherSpec withRC2393no FIPS. IBM MQ no garantiza rechazar todas las conexiones de este tipo y es responsabilidad del usuario determinar si la configuración es compatible con IBM MQ.

SCKR (entero con signo de 10 dígitos)

Este campo sólo es relevante para IBM MQ MQI clients que se ejecuta en sistemas UNIX y Windows . Especifica la ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. El archivo de base de datos de claves debe tener un nombre de archivo con el formato `zzz.kdb`, donde `zzz` es seleccionable por el usuario. El campo *SCKR* contiene la vía de acceso a este archivo, junto con la raíz del nombre de archivo (todos los caracteres del nombre de archivo hasta, pero sin incluir, el `.kdb`final). El sufijo de archivo `.kdb` se añade automáticamente.

Cada archivo de base de datos de claves tiene un *archivo de ocultación de contraseña*asociado. Esto contiene contraseñas cifradas que se utilizan para permitir el acceso programático a la base de datos de claves. El archivo de ocultación de contraseña debe residir en el mismo directorio y tener la misma raíz de archivo que la base de datos de claves, y debe finalizar con el sufijo `.sth`.

Por ejemplo, si el campo *SCKR* tiene el valor `/xxx/yyy/key`, el archivo de base de datos de claves debe ser `/xxx/yyy/key.kdb`y el archivo de ocultación de contraseña debe ser `/xxx/yyy/key.sth`, donde `xxx` y `yyy` representan nombres de directorio.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. El valor no se comprueba; si hay un error al acceder al repositorio de claves, la llamada falla con el código de razón RC2381.

Para ejecutar una conexión TLS desde un IBM MQ MQI client, establezca *SCKR* en un nombre de archivo de base de datos de claves válido.

Este es un campo de entrada. La longitud de este campo la proporciona LNSSKR. El valor inicial de este campo es un carácter en blanco.

SCSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

SCSIDV
Identificador para la estructura de opciones de configuración TLS.

Siempre es un campo de entrada. El valor inicial de este campo es SCSIDV.

SCVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SCVER1
Estructura de opciones de configuración de TLS Version-1 .

SCVER2
Estructura de opciones de configuración de TLS Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

SCVERC
Versión actual de la estructura de opciones de configuración de TLS.

Siempre es un campo de entrada. El valor inicial de este campo es SCVER2

Valores iniciales

Tabla 726. Valores iniciales de campos en MQSCO

Nombre de campo	Nombre de constante	Valor de constante
SCSID	SCSIDV	`'SCO¬'`
SCVER	SCVER5	1
SCKR	Ninguna	Serie nula o espacios en blanco

Tabla 726. Valores iniciales de campos en MQSCO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
SCCH	Ninguna	Serie nula o espacios en blanco
SCAIC	Ninguna	0
SCAIO	Ninguna	0
SCAIP	Ninguna	Puntero nulo o bytes nulos
SCKRC	Ninguna	Puntero nulo o bytes nulos
SCFR	Ninguna	Puntero nulo o bytes nulos
SCEPSUITEB	Ninguna	Puntero nulo o bytes nulos
SCCERTVPOL	Ninguna	Puntero nulo o bytes nulos
SCCERLBL	Ninguna	Puntero nulo o bytes nulos

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. Consulte"Declaración RPG" en la página 1240para ver las opciones de SCEPSUITEB .

Declaración 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 (Descriptor de suscripción) en IBM i

La estructura MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando.

Visión general

Finalidad

La estructura es un parámetro de entrada/salida en la llamada MQSUB.

Suscripciones gestionadas

Si una aplicación no tiene ninguna necesidad específica de utilizar una cola determinada como destino para las publicaciones que coinciden con su suscripción, puede utilizar la característica de suscripción gestionada. Si una aplicación opta por utilizar una suscripción gestionada, el gestor de colas informa al suscriptor sobre el destino donde se envían los mensajes publicados, proporcionando un descriptor de objeto como salida de la llamada MQSUB. Para obtener más información, consulte HOBJ (entero con signo de 10 dígitos)-entrada/salida.

Cuando se elimina la suscripción, el gestor de colas también se compromete a limpiar los mensajes que no se han recuperado del destino gestionado, en las situaciones siguientes:

- Cuando se elimina la suscripción-mediante el uso de MQCLOSE con CORMSB-y se cierra el Hobj gestionado.
- Por medios implícitos cuando se pierde la conexión con una aplicación utilizando una suscripción no duradera (SONDUR)
- Por caducidad cuando se elimina una suscripción porque ha caducado y el Hobj gestionado está cerrado.

Debe utilizar suscripciones gestionadas con suscripciones no duraderas, para que se pueda realizar la limpieza, y para que los mensajes de las suscripciones no duraderas cerradas no ocupen espacio en el gestor de colas. Las suscripciones duraderas también pueden utilizar destinos gestionados.

Juego de caracteres y codificación

Los datos de MQSD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionado por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la página 1241
- “Valores iniciales” en la página 1254
- “Declaración RPG” en la página 1255

Campos

La estructura MQSD contiene los campos siguientes; los campos se describen en orden alfabético:

SDAID (serie de caracteres de 32 bytes)

Este valor está en el campo *MDAID* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *SDAID* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para obtener más información sobre *MDAID* , consulte MDAID.

Si no se especifica la opción SOSETI , el *MDAID* que se establece en cada mensaje publicado para esta suscripción está en blanco, como información de contexto predeterminada.

Si se especifica la opción SOSETI , el usuario está generando el *SDAID* y este campo es un campo de entrada que contiene el *MDAID* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona LNAIDD. El valor inicial de este campo es de 32 caracteres en blanco.

Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar el *SDAID* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando SORES, este campo se establece en el *MDAID* actual que se está utilizando para la suscripción.

SDACC (serie de caracteres de 32 bytes)

Este valor está en el campo *MDACC* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *MDACC* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para obtener más información sobre *MDACC* , consulte MDACC.

Puede utilizar el siguiente valor especial para el campo *SDACC* :

ACNONE
No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

Si no se especifica la opción SOSETI , el gestor de colas genera la señal de contabilidad como información de contexto predeterminada y este campo es un campo de salida que contiene el *MDACC* que se establece en cada mensaje publicado para esta suscripción.

Si se especifica la opción SOSETI , el usuario está generando la señal de contabilidad y este campo es un campo de entrada que contiene el *MDACC* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona LNACCT. El valor inicial de este campo es ACNONE.

Si se modifica una suscripción existente utilizando la opción SOALT , el valor de *MDACC* en cualquier mensaje de publicación futuro se puede cambiar.

Al volver de una llamada MQSUB utilizando SORES, este campo se establece en el *MDACC* actual que se utiliza para la suscripción.

SDASI (serie de bits de 40 bytes)

Se trata de un identificador de seguridad que se pasa con *SDAU* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas.

SDASI sólo se utiliza si se especifica SOALTU y el campo *SDAU* no está completamente en blanco hasta el primer carácter nulo o el final del campo.

Al volver de una llamada MQSUB utilizando SORES, este campo no se modifica.

Consulte la descripción de ODASI en el tipo de datos MQOD para obtener más información.

SDAU (serie de caracteres de 12 bytes)

Si especifica SOALTU, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la suscripción y para la salida a la cola de destino (especificada en el parámetro **Hobj** de la llamada MQSUB), en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación.

Si es satisfactorio, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se ejecuta actualmente la aplicación.

Si se especifica SOALTU y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción sólo puede realizarse correctamente si no se necesita ninguna autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica SOALTU , este campo se ignora.

Al volver de una llamada MQSUB utilizando SORES, este campo no se modifica.

Este es un campo de entrada. La longitud de este campo la proporciona LNUID. El valor inicial de este campo es de 12 caracteres en blanco.

SDCID (serie de bits de 24 bytes)

Todas las publicaciones enviadas para que coincidan con esta suscripción contienen este identificador de correlación en el descriptor de mensaje. Si varias suscripciones utilizan la misma cola para obtener sus publicaciones, el uso de MQGET por ID de correlación sólo permite obtener publicaciones para una suscripción específica. Este identificador de correlación puede ser generado por el gestor de colas o por el usuario.

Si no se especifica la opción SOSCID , el gestor de colas genera el identificador de correlación y este campo es un campo de salida que contiene el identificador de correlación que se establece en cada mensaje publicado para esta suscripción.

Si se especifica la opción SOSCID , el usuario está generando el identificador de correlación y este campo es un campo de entrada que contiene el identificador de correlación que se debe establecer en cada publicación para esta suscripción. En este caso, si el campo contiene CINONE, el identificador de correlación que se establece en cada mensaje publicado para esta suscripción es el identificador de correlación que ha creado la colocación original del mensaje.

Si se especifica la opción SOGRP y el identificador de correlación especificado es el mismo que una suscripción agrupada existente que utiliza la misma cola y una serie de tema solapada, sólo se proporciona una copia de la publicación a la suscripción más significativa del grupo.

La longitud de este campo la proporciona LNCID. El valor inicial de este campo es CINONE.

Si se modifica una suscripción existente utilizando la opción SOALT , y este campo es un campo de entrada, el ID de correlación de suscripción se puede cambiar, a menos que la suscripción se haya creado utilizando la opción SOGRP .

Al volver de una llamada MQSUB utilizando SORES, este campo se establece en el ID de correlación actual para la suscripción.

SDEXP (entero con signo de 10 dígitos)

Este es el tiempo expresado en décimas de segundo tras el cual caduca la suscripción. No habrá más publicaciones que coincidan con esta suscripción después de que haya pasado este intervalo. También se utiliza como valor en el campo *MDEXP* en el MQMD de las publicaciones enviadas a este suscriptor.

Se reconoce el siguiente valor especial:

EIULIM
La suscripción tiene un tiempo de caducidad ilimitado.

Si se modifica una suscripción existente utilizando la opción SOALT , se puede cambiar la caducidad de la suscripción.

Al volver de una llamada MQSUB utilizando la opción SORES , este campo se establece en la caducidad original de la suscripción y no en el tiempo de caducidad restante.

SDON (serie de caracteres de 48 bytes)

Es el nombre del objeto de tema tal como se ha definido en el gestor de colas local.

El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el nulo y los caracteres que le siguen se tratan como espacios en blanco. Las restricciones son:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- Los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

SDON se utiliza para formar el nombre de tema completo.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *SDON* y *SDOS*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte Combinación de series de tema.

Al volver de una llamada MQSUB utilizando la opción SORES , este campo no se modifica.

La longitud de este campo la proporciona LNTOPN. El valor inicial de este campo es de 48 caracteres en blanco.

Si se modifica una suscripción existente utilizando la opción SDALT, el nombre del objeto de tema al que se ha suscrito no se puede cambiar. Este campo y *SDOS* se pueden omitir. Si se proporcionan, deben resolver el mismo nombre de tema completo o la llamada falla con `RC2510` .

SDOPT (entero con signo de 10 dígitos)

Debe especificar al menos una de las opciones siguientes:

- `SOALT`
- `SORES`
- `SOCRT`

Se pueden añadir los valores. No añada la misma constante más de una vez. La tabla muestra cómo puede combinar estas opciones: las combinaciones que no son válidas se anotan; cualquier otra combinación es válida.

Opciones de acceso o creación

Las opciones de acceso y creación controlan si se crea una suscripción o si se devuelve o modifica una suscripción existente. Debe especificar al menos una de estas opciones. La tabla muestra combinaciones válidas de opciones de acceso o creación.

Tabla 727. Combinaciones válidas de opciones de acceso y creación

Combinación de opciones	Notas
SOCRT	Crea una suscripción si no existe; falla si la suscripción existe.
SORES	Reanuda una suscripción existente, falla si no existe ninguna suscripción.
SOCRT + SORES	Crea una suscripción si no existe una y reanuda una que coincida, si existe. Combinación útil si se utiliza en una aplicación que puede ejecutarse varias veces.
SORES + SOALT (ver nota)	Reanuda una suscripción existente, alterando los campos para que coincidan con los especificados en MQSD, falla si no existe ninguna suscripción.

Tabla 727. Combinaciones válidas de opciones de acceso y creación (continuación)	
Combinación de opciones	**Notas**
SOCRT + SOALT (ver nota)	Crea una suscripción si no existe una y reanuda una coincidencia, si existe, alterando los campos para que coincidan con los especificados en el MQSD. Combinación útil si se utiliza en una aplicación que desea asegurarse de que su suscripción está en un estado determinado antes de continuar.

Nota:

Las opciones que especifican SOALT también pueden especificar SORES, pero esta combinación no tiene ningún efecto adicional para especificar solo SOALT . SOALT implica SORES, porque llamar a MQSUB para modificar una suscripción implica que las suscripciones también se reanudan. Sin embargo, lo contrario no es cierto: reanudar una suscripción no implica que deba ser alterada.

SOCRT

Cree una suscripción para el tema especificado. Si existe una suscripción que utiliza el mismo *SDSN* , la llamada falla con RC2432 . Esta anomalía se puede evitar combinando la opción SOCRT con SORES. *SDSN* no siempre es necesario. Para obtener más detalles, consulte la descripción de ese campo.

La combinación de SOCRT con SORES primero comprueba si hay una suscripción existente para el *SDSN*especificado, y si hay un descriptor de contexto para esa suscripción preexistente; pero si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en MQSD.

SOCRT también se puede combinar con SOALT con un efecto similar (consulte los detalles sobre SOALT más adelante en este tema).

SORES

Devuelve un descriptor de contexto a una suscripción preexistente que coincide con las especificadas por *SDSN*. No se realizan cambios en los atributos de suscripción coincidentes y se devuelven en la salida de la estructura MQSD. La mayoría del contenido de MQSD no se utiliza: los campos utilizados son *SDSID*, *SDVER*, *SDOPT*, *SDAID* y *SDASI*, y *SDSN*.

La llamada falla con el código de razón RC2428 si no existe una suscripción que coincida con el nombre de suscripción completo. Esta anomalía se puede evitar combinando la opción SOCRT con SORES. Para obtener detalles sobre SOCRT, consulte SOCRT.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción, o si posteriormente ha sido alterado por un ID de usuario diferente, es el ID de usuario de la modificación más reciente y satisfactoria. Si se utiliza un *SDAID* y se permite el uso de ID de usuario alternativos para dicho usuario, *SDAID* se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

El ID de usuario que ha creado la suscripción se registra como *SDAU* si se utiliza ese campo, y se permite el uso de ID de usuario alternativos para ese usuario.

Si existe una suscripción coincidente que se ha creado sin la opción SOAUID y el ID de usuario de la suscripción es diferente del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón RC2434 .

Si existe una suscripción coincidente y otra aplicación la está utilizando actualmente, la llamada falla con el código de razón RC2429 . Si está actualmente en uso por la misma conexión, la llamada no falla y se devuelve un descriptor de contexto a la suscripción.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con RC2523 .

SORES está implícito en SOALT y, por lo tanto, no es necesario combinarlo con esa opción, sin embargo, no es un error si se combinan estas dos opciones.

SOALT

Devuelve un descriptor de contexto a una suscripción preexistente con el nombre de suscripción completo que coincide con los especificados en *SDSN*. Los atributos de la suscripción que son diferentes de los especificados en MQSD se modifican en la suscripción a menos que la modificación no esté permitida para ese atributo. Los detalles se anotan en la descripción de cada atributo y se resumen en la tabla siguiente. Si intenta modificar un atributo que no se puede cambiar, la llamada falla con el código de razón que se muestra en la tabla siguiente.

La llamada falla con el código de razón RC2428 si no existe una suscripción que coincida con el nombre de suscripción completo. Esta anomalía se puede evitar combinando la opción SOCRT con SOALT.

La combinación de SOCRT con SOALT comprueba en primer lugar si hay una suscripción existente para el nombre de suscripción completo especificado, y si hay un descriptor de contexto para esa suscripción preexistente con las modificaciones realizadas como se ha detallado anteriormente; pero si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en MQSD.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción, o si posteriormente ha sido alterado por un ID de usuario diferente, es el ID de usuario de la modificación correcta más reciente. Si se utiliza *SDAU* (y se permite el uso de ID de usuario alternativos para ese usuario), el ID de usuario alternativo se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

Si existe una suscripción coincidente que se ha creado sin la opción SOAUID y el ID de usuario de la suscripción es diferente del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón RC2434 .

Si existe una suscripción coincidente y la está utilizando actualmente otra aplicación, la llamada falla con RC2429 . Si está actualmente en uso por la misma conexión, la llamada no falla y se devuelve un descriptor de contexto a la suscripción.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con RC2523 .

Las tablas siguientes muestran los atributos de suscripción que SOALTpuede modificar.

Tabla 728. Atributos en MQSD y MQSUB que se pueden modificar

Descriptor de tipo de datos o llamada de función	Nombre de campo	¿Se puede modificar este atributo utilizando SOALT?	Código de razón
MQSD	Opciones de durabilidad	No	RC2509
MQSD	Opciones de destino	Sí	Ninguna
MQSD	Opciones de registro	Sí (consulte la nota 1)	RC2515 si intenta modificar SOGRP
MQSD	Opciones de publicación	Sí (consulte la nota 2)	Ninguna
MQSD	Opciones de comodín	No	RC2510
MQSD	Otras opciones	No (consulte la nota 3)	Ninguna
MQSD	ObjectName	No	RC2510
MQSD	SDAU	No (consulte la nota 4)	Ninguna
MQSD	SDASI	No (consulte la nota 4)	Ninguna
MQSD	SDEXP	Sí	Ninguna

Tabla 728. Atributos en MQSD y MQSUB que se pueden modificar (continuación)			
Descriptor de tipo de datos o llamada de función	**Nombre de campo**	**¿Se puede modificar este atributo utilizando SOALT?**	**Código de razón**
MQSD	SDOS	No	RC2510
MQSD	SDSN	No (consulte la nota 5)	Ninguna
MQSD	SDSUD	Sí	Ninguna
MQSD	SCID	Sí (consulte la nota 6)	RC2515 cuando está en una suscripción agrupada
MQSD	SDPRI	Sí	Ninguna
MQSD	SACC	Sí	Ninguna
MQSD	ID de SDA	Sí	Ninguna
MQSD	SDSL	No	RC2512
MQSUB	Hobj	Sí (consulte la nota 6)	RC2515 cuando está en una suscripción agrupada

Notas:

1. SOGRP no se puede modificar.
2. SONEWP no se puede modificar porque no forma parte de la suscripción
3. Estas opciones no forman parte de la suscripción
4. Este atributo no forma parte de la suscripción
5. Este atributo es la identidad de la suscripción que se está alterando
6. Alterable excepto cuando forma parte de una subagrupación (SOGRP)

Opciones de durabilidad: Las opciones siguientes controlan la durabilidad de la suscripción. Sólo puede especificar una de estas opciones. Si está alterando una suscripción existente utilizando la opción SOALT , no puede cambiar la durabilidad de la suscripción. Al volver de una llamada MQSUB utilizando SORES, se establece la opción de durabilidad adecuada.

SODUR
Solicite que la suscripción a este tema permanezca hasta que se elimine explícitamente utilizando MQCLOSE con la opción CORMSB . Si esta suscripción no se elimina explícitamente, permanecerá incluso después de que esta aplicación se conecte al gestor de colas.

Si se solicita una suscripción duradera a un tema que está definido como que no permite suscripciones duraderas, la llamada falla con RC2436 .

SONDUR
Solicite que la suscripción a este tema se elimine cuando se cierre la conexión de la aplicación con el gestor de colas, si todavía no se ha eliminado explícitamente. SONDUR es lo contrario de la opción SODUR y se define para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

Opciones de destino: Las opciones siguientes controlan el destino al que se envían las publicaciones de un tema al que se ha suscrito. Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar el destino utilizado para las publicaciones de la suscripción. Al volver de una llamada MQSUB utilizando SORES, esta opción se establece si procede.

SOMAN

Solicite que el gestor de colas gestione el destino al que se envían las publicaciones.

El descriptor de contexto de objeto devuelto en *HOBJ* representa una cola gestionada del gestor de colas y se utiliza con las llamadas MQGET, MQCB, MQINQ o MQCLOSE posteriores.

No se puede proporcionar un descriptor de contexto de objeto devuelto desde una llamada MQSUB anterior en el parámetro **Hobj** cuando no se especifica SOMAN .

Opciones de registro: Las opciones siguientes controlan los detalles del registro que se realiza en el gestor de colas para esta suscripción. Si se modifica una suscripción existente utilizando la opción SOALT , estas opciones de registro se pueden cambiar. Al volver de una llamada MQSUB utilizando SORES , se establecen las opciones de registro adecuadas.

SOGRP

Esta suscripción se agrupa con otras suscripciones del mismo *SDSL* utilizando la misma cola y especificando el mismo ID de correlación de modo que cualquier publicación a temas que pueda hacer que se proporcione más de un mensaje de publicación al grupo de suscripciones, debido a que se está utilizando un conjunto solapado de series de tema, sólo hace que se entregue un mensaje a la cola. Si no se utiliza esta opción, cada suscripción exclusiva (identificada por *SDSN*) que coincida se proporciona con una copia de la publicación, lo que puede significar que se pueda colocar más de una copia de la publicación en la cola compartida por un número de suscripciones.

Sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación. La suscripción más significativa se basa en el nombre de tema completo hasta el punto en el que se encuentra un comodín. Si se utiliza una mezcla de esquemas de comodín dentro del grupo, sólo es importante la posición del comodín. Se recomienda no combinar diferentes esquemas de comodín dentro de un grupo de suscripciones que comparten la misma cola.

Al crear una nueva suscripción agrupada, todavía debe tener un *SDSN*exclusivo, pero si coincide con el nombre de tema completo de una suscripción existente en el grupo, la llamada falla con RC2514 .

Si la suscripción más significativa del grupo también especifica SONOLC y se trata de una publicación de la misma aplicación, no se entrega ninguna publicación a la cola.

Al modificar una suscripción realizada con esta opción, los campos que implican la agrupación, *Hobj* en la llamada MQSUB (que representa la cola y el nombre del gestor de colas) y *SDCID* no se pueden cambiar. El intento de modificarlos hace que la llamada falle con RC2515 .

Esta opción se debe combinar con SOSCID con un *SDCID* que no esté establecido en CINONEy no se pueda combinar con SOMAN.

SAUID

Cuando se especifica SOAUID , la identidad del suscriptor no está restringida a un único ID de usuario. Esto permite que cualquier usuario modifique o reanude la suscripción cuando disponga de la autoridad adecuada. Sólo un único usuario puede tener la suscripción a la vez. Un intento de reanudar el uso de una suscripción actualmente en uso por otra aplicación hace que la llamada falle con RC2429 .

Para añadir esta opción a una suscripción existente, la llamada MQSUB, utilizando SOALT, debe proceder del mismo ID de usuario que la propia suscripción original.

Si una llamada MQSUB hace referencia a una suscripción existente con SOAUID establecido, y el ID de usuario difiere de la suscripción original, la llamada sólo se realiza correctamente si el nuevo ID de usuario tiene autorización para suscribirse al tema. Al finalizar correctamente, las futuras publicaciones de este suscriptor se colocan en la cola del suscriptor con el nuevo ID de usuario establecido en el mensaje de publicación.

No especifique SOAUID y SOFUID. Si no se especifica ninguno, el valor predeterminado es SOFUID.

SOFUID

Cuando se especifica SOFUID , sólo el último ID de usuario que modifica la suscripción puede modificar o reanudar la suscripción. Si la suscripción no se ha modificado, es el ID de usuario que ha creado la suscripción.

Si un verbo MQSUB hace referencia a una suscripción existente con SOAUID establecido y altera la suscripción utilizando SOALT para utilizar el SOFUID, el ID de usuario de la suscripción se fija ahora en este nuevo ID de usuario. La llamada sólo es satisfactoria si el nuevo ID de usuario tiene autoridad para suscribirse al tema.

Si un ID de usuario distinto del registrado como propietario de una suscripción intenta reanudar o modificar una suscripción SOFUID , la llamada falla con RC2434 . El ID de usuario propietario de una suscripción se puede visualizar utilizando el mandato **DISPLAY SBSTATUS** .

No especifique SOAUID y SOFUID. Si no se especifica ninguno, el valor predeterminado es SOFUID.

Opciones de publicación: Las opciones siguientes controlan la forma en que se envían las publicaciones a este suscriptor. Si se modifica una suscripción existente utilizando la opción SOALT , estas opciones de publicación se pueden cambiar.

SONOLC
Indica al intermediario que la aplicación no desea ver ninguna de sus propias publicaciones. Se considera que las publicaciones se han originado en la misma aplicación si los manejadores de conexión son los mismos. Al volver de una llamada MQSUB utilizando SORES , esta opción se establece si procede.

SONEWP
No se enviarán publicaciones retenidas actualmente, cuando se cree esta suscripción, sólo se enviarán nuevas publicaciones. Esta opción sólo se aplica cuando se especifica SOCRE . Los cambios posteriores en una suscripción no alteran el flujo de publicaciones y, por lo tanto, las publicaciones que se han retenido en un tema, ya se han enviado al suscriptor como nuevas publicaciones.

Si esta opción se especifica sin SOCRE , hace que la llamada falle con RC2046 . Al volver de una llamada MQSUB utilizando SORES , esta opción no se establece incluso si la suscripción se ha creado utilizando esta opción.

Si no se utiliza esta opción, los mensajes retenidos anteriormente se envían a la cola de destino proporcionada. Si esta acción falla debido a un error, ya sea RC2525 o RC2526 , la creación de la suscripción falla.

Esta opción no es válida en combinación con SOPUBR.

SOPUBR
El establecimiento de esta opción indica que el suscriptor solicita información específicamente cuando es necesario. El gestor de colas no envía mensajes no solicitados al suscriptor. La publicación retenida (o posiblemente varias publicaciones si se especifica un comodín en el tema) se envía al suscriptor cada vez que se realiza una llamada MQSUBRQ utilizando el descriptor de contexto Hsub de una llamada MQSUB anterior. No se envían publicaciones como resultado de la llamada MQSUB utilizando esta opción. Al volver de una llamada MQSUB utilizando SORES , esta opción se establece si procede.

Esta opción no es válida en combinación con SONEWP.

Opciones de comodín: Las opciones siguientes controlan cómo se interpretan los comodines en la serie proporcionada en el campo *SDOS* de MQSD. Sólo puede especificar una de estas opciones. Si se modifica una suscripción existente utilizando la opción SOALT , estas opciones de comodín no se pueden cambiar. Al volver de una llamada MQSUB utilizando SORES , se establece la opción de comodín adecuada.

SOCHR
Los comodines sólo operan en caracteres dentro de la serie de tema. El campo SOWCHR trata la barra inclinada (/) como un carácter más sin ninguna significación especial.

El comportamiento definido por SOWCHR se muestra en la tabla siguiente:

Tabla 729. Cómo se interpretan los comodines	
Carácter especial	**Comportamiento**
*	Comodín, cero o más caracteres
?	Comodín, un carácter
%	Carácter de escape para permitir que los caracteres '*', '?' o '%' se utilicen en una serie y no se interpreten como un carácter especial, por ejemplo, '% *', '%?' o' %% '.

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Nota: Este uso de comodines proporciona exactamente el significado proporcionado en IBM MQ V6 y WebSphere MB V6 cuando se utilizan mensajes con formato MQRFH1 para publicación/suscripción. Se recomienda que no se utilice para las aplicaciones recién escritas y que solo se utilice para las aplicaciones que se ejecutaban anteriormente en esa versión y que no se han modificado para utilizar el comportamiento de comodín predeterminado tal como se describe en SOWTOP.

SOWTOP

Los comodines sólo operan en elementos de tema dentro de la serie de tema. Este es el comportamiento predeterminado si no se elige ninguno.

El comportamiento necesario para SOWTOP se muestra en la tabla siguiente:

Tabla 730. Cómo se interpretan los comodines	
Carácter especial	**Comportamiento**
/	Separador de nivel de tema
#	Comodín: nivel de varios temas
+	Comodín: nivel de tema único

Nota:

Los caracteres '+' y '#' no se tratan como comodines si se mezclan con otros caracteres (incluidos ellos mismos) dentro de un nivel de tema. En la serie siguiente, los caracteres '#' y '+' se tratan como caracteres ordinarios.

```
level0/level1/#+/level3/level#
```

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Nota: Este uso de comodines proporciona el significado que se proporciona en WebSphere Message Broker 6 cuando se utilizan mensajes con formato MQRFH2 para la publicación/suscripción.

Otras opciones: Las opciones siguientes controlan la forma en que se emite la llamada de API en lugar de la suscripción. Al volver de una llamada MQSUB utilizando SORES , estas opciones no se modifican.

SOALTU
El campo SDAU contiene un identificador de usuario que se debe utilizar para validar esta llamada MQSUB. La llamada sólo puede tener éxito si esta SDAU está autorizada a abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado a hacerlo.

SOSCID
La suscripción debe utilizar el identificador de correlación proporcionado en el campo *SDCID* . Si no se especifica esta opción, el gestor de colas crea automáticamente un identificador de correlación en el momento de la suscripción y se devuelve a la aplicación en el campo *SDCID* . Consulte SDCID (serie de 24 bytes) SDCID para obtener más información.

SOSETI

La suscripción es para utilizar la señal de contabilidad y los datos de identidad de aplicación proporcionados en los campos *SDACC* y *SDAID* .

Si se especifica esta opción, se lleva a cabo la misma comprobación de autorización que si se accediera a la cola de destino utilizando una llamada MQOPEN con OOSETI, excepto en el caso en que también se utilice la opción SOMAN , en cuyo caso no habrá ninguna comprobación de autorización en la cola de destino.

Si no se especifica esta opción, las publicaciones enviadas a este suscriptor tienen asociada la información de contexto predeterminada:

Tabla 731. Información de contexto predeterminada para las publicaciones enviadas a este suscriptor

Campo de MQMD	Valor utilizado
MDUID	El ID de usuario asociado a la suscripción en el momento en que se realizó la suscripción.
MDACC	Se determina a partir del entorno, si es posible; se establece en ACNONE si no es así.
MDAID	Establecer en blancos

Esta opción sólo es válida con SOCRE y SOALT. Si se utiliza con SORES, los campos *SDACC* y *SDAID* se ignoran, por lo que esta opción no tiene ningún efecto.

Si se modifica una suscripción sin utilizar esta opción en la que previamente la suscripción había facilitado información de contexto de identidad, se genera información del contexto predeterminado para la suscripción modificada.

Si una suscripción que permite que distintos ID de usuario la utilicen con la opción SOAUID, se reanuda mediante un ID de usuario diferente, se genera el contexto de identidad predeterminado para el nuevo ID de usuario que ahora es propietario de la suscripción y se entregan las publicaciones posteriores que contengan el nuevo contexto de identidad.

SOFIQ
La llamada MQSUB falla si el gestor de colas está en estado de desactivación temporal. En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQSUB falle si la conexión está en estado de desactivación temporal.

SDAU (serie de caracteres de 12 bytes)

Si especifica SOALTU, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la suscripción y para la salida a la cola de destino (especificada en el parámetro **Hobj** de la llamada MQSUB), en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación.

Si es satisfactorio, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se ejecuta actualmente la aplicación.

Si se especifica SOALTU y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción sólo puede realizarse correctamente si no hay autorización de usuario que deba suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica SOALTU , este campo se ignora.

Al volver de una llamada MQSUB utilizando SORES, este campo no se modifica.

Este es un campo de entrada. La longitud de este campo la proporciona LNUID. El valor inicial de este campo es de 12 caracteres en blanco.

SDPRI (entero con signo de 10 dígitos)

Es el valor que se encuentra en el campo *MQPRI* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. Para obtener más información sobre el campo *MQPRI* en MQMD, consulte MDPRI.

El valor debe ser mayor o igual que cero, donde cero es la prioridad más baja. También se pueden utilizar los valores especiales siguientes:

PRQDEF

Cuando se proporciona una cola de suscripción en el campo Hobj de la llamada MQSUB, y no es un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo **DefPriority** de esta cola. Si la cola así identificada es una cola de clúster o hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad se determina cuando el mensaje de publicación se coloca en la cola tal como se describe para MDPRI.

Si la llamada MQSUB utiliza un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo **DefPriority** de la cola modelo asociada al tema al que se ha suscrito.

PRPUB
La prioridad del mensaje es la prioridad de la publicación original. Es el valor inicial del campo.

Si se modifica una suscripción existente utilizando la opción SOALT , se puede cambiar el *MQPRI* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando SORES, este campo se establece en la prioridad actual que se utiliza para la suscripción.

SDRO (MQCHARV)

SDRO es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en *SDON*.

Si el nombre de objeto largo se proporciona en *SDOS* y no se proporciona nada en *SDON*, el valor devuelto en este campo es el mismo que el proporcionado en *SDOS*.

Si se omite este campo (es decir, SDRO.VSBufSize es cero), no se devuelve *SDRO* , pero la longitud se devuelve en SDRO.VSLength. Si la longitud es más corta que el *SDRO*completo, se trunca y devuelve tantos de los caracteres situados más a la derecha como pueda caber en la longitud proporcionada.

Si *SDRO* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2520 .

SDSID (serie de caracteres de 4 bytes)

Este es el identificador de estructura; el valor debe ser:

SDSIDV
Identificador de la estructura del descriptor de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es SDSIDV

SDSL (entero con signo de 10 dígitos)

Éste es el nivel asociado a la suscripción. Las publicaciones sólo se entregan a esta suscripción si está en el conjunto de suscripciones con el valor *SDSL* más alto menor o igual que el PubLevel utilizado en el momento de la publicación.

El valor debe estar en el rango de cero a 9. Cero sería el nivel más bajo.

El valor inicial de este campo es 1.

Si se modifica una suscripción existente utilizando la opción SOALT , *SDSL* no se puede cambiar.

SDSN (MQCHARV)

SDSN especifica el nombre de suscripción.

Este campo sólo es necesario si *SDOPT* especifica la opción SODUR , pero si se proporciona también lo utiliza el gestor de colas para SONDUR . Si se especifica, *SDSN* debe ser exclusivo dentro del gestor de colas, porque es el campo utilizado para identificar suscripciones.

La longitud máxima de *SDSN* es 10240.

Este campo sirve para dos propósitos. Para una suscripción SODUR, es el medio por el que identifica una suscripción para reanudarla después de que se haya creado, si ha cerrado el descriptor de contexto para la suscripción (utilizando la opción COKPSB) o se ha desconectado del gestor de colas. La identificación de una suscripción para eliminarla después de que se haya creado se realiza utilizando la llamada MQSUB con la opción SORES . El campo SDSN también se visualiza en la vista de administración de suscripciones en el campo *SDSN* en DISPLAY SBSTATUS.

Si *SDSN* se especifica incorrectamente, de acuerdo con la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, o si se omite cuando es necesario (es decir, *SDSN*. *VCHRL* es cero), o si supera la longitud máxima, la llamada falla con el código de razón RC2440 .

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción SOALT , el nombre de suscripción no se puede cambiar, porque es el campo utilizado para identificar la suscripción. No se cambia en la salida de una llamada MQSUB con la opción SORES .

SDSS (MQCHARV)

SDSS es la serie que proporciona los criterios de selección utilizados al suscribirse a mensajes de un tema.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción SORES , si se proporciona un almacenamiento intermedio, y si también hay una longitud de almacenamiento intermedio positiva en VSBufSize. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devuelve la longitud de la serie de selección en el campo VSLength de MQCHARV. Si el búfer proporcionado es inferior al espacio necesario para devolver el campo, solo se devuelven VSBufSize bytes en el búfer.

Si *SDSS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2519 .

SDSUD (MQCHARV)

Los datos proporcionados en la suscripción en este campo se incluyen como la propiedad de mensaje mq.SubUserData de cada publicación enviada a esta suscripción.

La longitud máxima de *SDSUD* es 10240.

Si *SDSUD* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón RC2431.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción SOALT , los datos de usuario de suscripción se pueden cambiar.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción SORES , si se proporciona un almacenamiento intermedio y hay una longitud de almacenamiento intermedio positiva en *VSBufLen*. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devuelve la longitud de los datos de usuario de suscripción en el campo *VCHRL* de MQCHARV. Si el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven *VSBufLen* bytes en el almacenamiento intermedio proporcionado.

SDVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SDVER1
Version-1 Estructura del descriptor de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

SDVERC
Versión actual de la estructura del descriptor de suscripción.

Siempre es un campo de entrada. El valor inicial del campo es SDVER1.

Valores iniciales

Tabla 732. Valores iniciales de los campos en MQSD

Nombre de campo	Nombre de constante	Valor de constante
SDSID	SDSIDV	'SD¬¬'
SDVER	SDVER1	1
SDOPT	SONDUR	0
SDON	Ninguna	Espacios en blanco
SDAU	Ninguna	Espacios en blanco
SDASI	SINONA	Nulos
SDEXP	EIULIM	-1
SDOS	Nombres y valores definidos para MQCHARV	
SDSN	Nombres y valores definidos para MQCHARV	
SDSUD	Nombres y valores definidos para MQCHARV	
SDCID	CINONA	Nulos

Tabla 732. Valores iniciales de los campos en MQSD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
SDPRI	PRQDEF	-3
SDACC	ACNONE	Nulos
SDAID	Ninguna	Espacios en blanco
SDSL	Ninguna	1
SDRO	Nombres y valores tal como se definen en MQCHARV	
Nota: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración 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 (Establecer opciones de propiedad de mensaje) en IBM i

La estructura **MQSMPO** permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes.

Visión general

Finalidad: la estructura es un parámetro de entrada en la llamada **MQSETMP** .

Conjunto de caracteres y codificación: los datos de **MQSMPO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- "Campos" en la página 1256
- "Valores iniciales" en la página 1257
- "Declaración RPG" en la página 1257

Campos

La estructura MQSMPO contiene los campos siguientes; los campos se describen en **orden alfabético**:

SPOPT (entero con signo de 10 dígitos)

Opciones de ubicación: Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de propiedad:

SSETF
Establece el valor de la primera propiedad que coincide con el nombre especificado, o si no existe, añade una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente.

SPSETC
Establece el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción IPINQF o IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, o cuando se especifica el descriptor de mensaje en el campo *HMSG* de la estructura MQGMO en una llamada MQGET o la estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando el cursor de propiedad todavía no se ha establecido o si la propiedad a la que apunta el cursor de propiedad se ha suprimido, la llamada falla con el código de terminación CCFAIL y el código de razón RC2471.

SSETA
Establece una nueva propiedad después de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción IPINQF o IPINQO.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, o cuando se especifica el descriptor de mensaje en el campo *HMSG* de la estructura MQGMO en una llamada MQGET o la estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando el cursor de propiedad todavía no se ha establecido o si la propiedad a la que apunta el cursor de propiedad se ha suprimido, la llamada falla con el código de terminación CCFAIL y el código de razón RC2471.

Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

SNONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es SPSETF.

SPSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

SPSIDV

Identificador para establecer la estructura de opciones de propiedad de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es **SPSIDV**.

SPVAKCSI (entero con signo de 10 dígitos)

El juego de caracteres del valor de propiedad que se va a establecer si el valor es una serie de caracteres.

Siempre es un campo de entrada. El valor inicial de este campo es **CSAPL**.

SPVALENC (entero con signo de 10 dígitos)

La codificación del valor de propiedad que se va a establecer si el valor es numérico.

Siempre es un campo de entrada. El valor inicial de este campo es **ENNAT**.

SPVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SPVER1
Version-1 establece la estructura de opciones de propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

SPVERC
Versión actual de la estructura de opciones de establecer propiedad de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es **SPVER1**.

Valores iniciales

Tabla 733. Valores iniciales de campos en MQSMPO

Nombre de campo	Nombre de constante	Valor de constante
SPSID	SPSIDV	'SMPO'
SPVER	SPVER1	1
SPOPT	SNONE	0
SPVALENC	ENNAT	Depende del entorno
SPVALCSI	CSAPL	-3

Declaración 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 (Opciones de solicitud de suscripción) en IBM i

La estructura MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción.

Visión general

Finalidad: la estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Versión: La versión actual de MQSRO es SRVER1.

- "Campos" en la página 1258
- "Valores iniciales" en la página 1259
- "Declaración RPG" en la página 1259

Campos

La estructura MQSRO contiene los campos siguientes; los campos se describen en **orden alfabético**:

SRNMP (entero con signo de 10 dígitos)

Es un campo de salida, devuelto a la aplicación para indicar el número de publicaciones enviadas a la cola de suscripción como resultado de esta llamada. Aunque este número de publicaciones se han enviado como resultado de esta llamada, no hay garantía de que este número de mensajes estén disponibles para que la aplicación los obtenga, especialmente si son mensajes no persistentes.

Puede haber más de una publicación si el tema al que se ha suscrito contenía un comodín. Si no había comodines en la serie de tema cuando se creó la suscripción representada por *HSUB* , como máximo se envía una publicación como resultado de esta llamada.

SROPT (entero con signo de 10 dígitos)

Debe especificarse una de las opciones siguientes. Sólo se puede especificar una opción.

Otras opciones: La siguiente opción controla lo que sucede cuando el gestor de colas se está desactivando temporalmente:

SRFIQ

La llamada MQSUBRQ falla si el gestor de colas está en estado de desactivación temporal.

Opción predeterminada: Si la opción descrita anteriormente no es necesaria, se debe utilizar la opción siguiente:

SRNONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

SRNONE ayuda a la documentación del programa. Aunque no se pretende que esta opción se utilice con ninguna otra, debido a que su valor es cero, no se puede detectar este uso.

SRSID (serie de caracteres de 4 bytes)

Este es el identificador de estructura; el valor debe ser:

SRSIDV
Identificador de la estructura SROPT de solicitud de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es SRSIDV.

SRVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SRVER1
Version-1 Estructura de opciones de solicitud de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

SRVERC
Versión actual de la estructura de opciones de solicitud de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es SRVER1.

Valores iniciales

Tabla 734. Valores iniciales de los campos en MQSRO

Nombre de campo	Nombre de constante	Valor de constante
SRSID	SRSIDV	'SRO¬'
SRVER	SRVER1	1
SROPT	SRNONE	0
SRNMP	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

Declaración 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 (estructura de informes de estado) en IBM i

La estructura MQSTS describe los datos en la estructura de estado devuelta por el mandato MQSTAT.

Visión general

Juego de caracteres y codificación: los datos de caracteres en MQSTS están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQSTS están en la codificación de máquina nativa; esto lo proporciona *ENNAT*.

Uso: El mandato MQSTAT se utiliza para recuperar información de estado. Esta información se devuelve en una estructura MQSTS. Para obtener información sobre MQSTAT, consulte “MQSTAT (Recuperar información de estado) en IBM i” en la página 1392.

- “Campos” en la página 1260
- “Valores iniciales” en la página 1263

- “Declaración RPG” en la página 1263

Campos

La estructura MQSTS contiene los campos siguientes; los campos se describen en **orden alfabético**:

STSCC (entero con signo de 10 dígitos)

Es el código de terminación resultante del primer error notificado en la estructura MQSTS.

Siempre es un campo de salida. El valor inicial de este campo es CCOK.

STSFC (entero con signo de 10 dígitos)

Es el número de llamadas de colocación asíncronas que han fallado.

Se trata de un campo de salida. El valor inicial de este campo es 0.

STSOBJN (serie de caracteres de 48 bytes)

Es el nombre local del objeto implicado en la primera anomalía.

Se trata de un campo de salida. El valor inicial de este campo es de 48 caracteres en blanco.

STSOQMGR (serie de caracteres de 48 bytes)

Es el nombre del gestor de colas en el que se define el objeto *STSOBJN* . Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Se trata de un campo de salida. El valor inicial de este campo es de 48 caracteres en blanco.

STSOO (entero con signo de 10 dígitos)

El STSOO utilizado para abrir el objeto sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

El valor de STSOO depende del valor del parámetro MQSTAT **STYPE** .

STATAPT

Cero.

STATREC

Cero.

STATRER

El STSOO utilizado cuando se produjo la anomalía. La razón de la anomalía se notifica en los campos *STSCC* y *STSRC* de la estructura MQSTS .

STSOO es un campo de salida. Su valor inicial es cero.

STSOS (MQCHARV)

Nombre de objeto largo del objeto anómalo sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

STSOS es un campo MQCHARV con una longitud máxima de 10240. Consulte MQCHARV para obtener una descripción de cómo utilizar la estructura MQCHARV.

La interpretación de STSOS depende del valor del parámetro MQSTAT **STYPE** .

STATAPT

Es el nombre de objeto largo de la cola o tema utilizado en la operación MQPUT , que ha fallado.

STATREC

Serie de caracteres de longitud cero

STATRER

Es el nombre de objeto largo del objeto que ha hecho que fallara la reconexión.

STS0S es un campo de salida. Su valor inicial es una serie de longitud cero.

STSOT (entero con signo de 10 dígitos)

El tipo de objeto que se denomina en *ObjectName*. Los valores posibles son:

OTALSQ
Cola alias.

OTLOCQ
Cola local.

OTMODQ
Cola modelo.

OTQ
Cola.

OTREMQ
Cola remota.

OTTOP
.

Siempre es un campo de salida. El valor inicial de este campo es OTQ.

STSRC (entero con signo de 10 dígitos)

Este es el código de razón resultante del primer error notificado en la estructura MQSTS

Siempre es un campo de salida. El valor inicial de este campo es RCNONE.

STSROBJN (serie de caracteres de 48 bytes)

Es el nombre de la cola de destino indicada en *STSOBJN* después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *STSRQMGR*.

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *STSROBJN* se establece en blancos:

- Un tema
- Una cola, pero no abierta para examinar, entrar o salir

Se trata de un campo de salida. El valor inicial de este campo es de 48 caracteres en blanco.

STSRQMGR (serie de caracteres de 48 bytes)

Es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *STSROBJN*. *STSRQMGR* puede ser el nombre del gestor de colas local.

Si *STSROBJN* es una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *STSRQMGR* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *STSROBJN* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *STSRQMGR* se establece en blancos:

- Un tema
- Una cola, pero no abierta para examinar, entrar o salir

- Una cola de clúster con OOBNDN especificado (o con OOBNDQ en vigor cuando el atributo de cola **DefBind** tiene el valor OOBNDN)

Se trata de un campo de salida. El valor inicial de este campo es de 48 caracteres en blanco.

STSSC (entero con signo de 10 dígitos)

Es el número de llamadas de colocación asíncronas que se han realizado correctamente.

Se trata de un campo de salida. El valor inicial de este campo es 0.

STSSID (serie de caracteres de 4 bytes)

Es el identificador de estructura. El valor debe ser:

IDSTSS
Identificador de la estructura de informes de estado.

El valor inicial de este campo es STSSID.

STSSO (entero con signo de 10 dígitos)

El STSSO utilizado para abrir la suscripción anómala. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de STSSO depende del valor del parámetro MQSTAT **STYPE** .

STATAPT

Cero.

STATREC

Cero.

STATRER

El STSSO utilizado cuando se produjo la anomalía. La razón de la anomalía se notifica en los campos *STSCC* y *STSRC* de la estructura MQSTS . Si la anomalía no está relacionada con la suscripción a un tema, el valor devuelto es cero.

STSSO es un campo de salida. Su valor inicial es cero.

STSSUN (MQCHARV)

El nombre de la suscripción anómala. Sólo está presente en la versión 2 de MQSTS o superior.

STSSUN es un campo MQCHARV con una longitud máxima de 10240. Consulte MQCHARV para obtener una descripción de cómo utilizar la estructura MQCHARV.

La interpretación de STSSUN depende del valor del parámetro MQSTAT **STYPE** .

STATAPT

Serie de longitud cero.

STATREC

Serie de longitud cero.

STATRER

El nombre de la suscripción que ha hecho que fallara la reconexión. Si no hay ningún nombre de suscripción disponible, o la anomalía no está relacionada con una suscripción, se trata de una serie de longitud cero.

STSSUN es un campo de salida. Su valor inicial es una serie de longitud cero.

STSVER (entero con signo de 10 dígitos)

Es el número de versión de la estructura. El valor debe ser:

STSVR1
Número de versión para la estructura de informes de estado.

La constante siguiente especifica el número de versión de la versión actual:

STSVR
Versión actual de la estructura de informes de estado.

El valor inicial de este campo es STSVR1.

STSWC (entero con signo de 10 dígitos)

Es el número de llamadas de colocación asíncronas que se han completado con un aviso.

Se trata de un campo de salida. El valor inicial de este campo es 0.

Valores iniciales

Tabla 735. Valores iniciales de campos en MQSTS

Nombre de campo	Nombre de constante	Valor de constante
STSSID	STSID	
STSVER	STSVR	STSVR1
STSCC	CCOK	0
STSRC	RCNONE	0
STSSC	Ninguna	0
STSWC	Ninguna	0
STSFC	Ninguna	0
STSOT	Ninguna	0
STSOBJN	Ninguna	Espacios en blanco
STSOQMGR	Ninguna	Espacios en blanco
STSROBJN	Ninguna	Espacios en blanco
STSRQMGR	Ninguna	Espacios en blanco
STSOS	Nombres y valores definidos para MQCHARV	
STSSUN	Nombres y valores definidos para MQCHARV	
STSOO	Ninguna	0
STSSO	Ninguna	0

Declaración 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-Mensaje de desencadenante

La estructura MQTM describe los datos del mensaje desencadenante que envía el gestor de colas a una aplicación de supervisor desencadenante cuando se produce un suceso desencadenante para una cola.

Visión general

Finalidad: esta estructura forma parte de la interfaz de supervisor desencadenante (TMI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ .

Nombre de formato: FMTM.

Conjunto de caracteres y codificación: los datos de caracteres de MQTM se encuentran en el juego de caracteres del gestor de colas que genera MQTM. Los datos numéricos en MQTM están en la codificación de máquina del gestor de colas que genera MQTM.

El juego de caracteres y la codificación de MQTM se proporcionan mediante los campos *MDCSI* y *MDENC* en:

- El MQMD (si la estructura MQTM está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQTM (todos los demás casos).

Uso: es posible que una aplicación de supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación iniciada por la aplicación de supervisor desencadenante. La información que puede necesitar la aplicación iniciada incluye *TMQN*, *TMTD*y *TMUD*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada. Para obtener información sobre MQTMC2, consulte “MQTMC2 (Formato de 2 caracteres de mensaje desencadenante) en IBM i” en la página 1269.

- En IBM i, la aplicación de supervisor desencadenante proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Para obtener información sobre los desencadenantes, consulte Requisitos previos para el desencadenamiento.

- “MQMD para un mensaje desencadenante” en la página 1265
- “Campos” en la página 1266
- “Valores iniciales” en la página 1268
- “Declaración RPG” en la página 1268

MQMD para un mensaje desencadenante

Tabla 736. Valores para los campos en el MQMD de un mensaje desencadenante generado por el gestor de colas

Campo de MQMD	Valor utilizado
MDSID	MDSIDV
MDVER	MDVER1
MDREP	RONONA
MDMT	MTDGRM
MDEXP	EIULIM
MDFB	FBNONE
MDENC	ENNAT
MDCSI	Atributo **CodedCharSetId** del gestor de colas
MDFMT	FMTM
MDPRI	Atributo **DefPriority** de la cola de inicio
MDPER	PENDIENTE
MDMID	Un valor exclusivo
MDCID	CINONA
MDBOC	0
MDRQ	Espacios en blanco
MDRM	Nombre de gestor de colas
MDUID	Espacios en blanco
MDACC	ACNONE
MDAID	Espacios en blanco
MDPAT	ATQM, o según corresponda para el agente de canal de mensajes
MDPAN	Primeros 28 bytes del nombre del gestor de colas
MDPD	Fecha en la que se envía el mensaje desencadenante
MDPT	Hora a la que se envía el mensaje desencadenante
MDAOD	Espacios en blanco

Se recomienda una aplicación que genere un mensaje desencadenante para establecer valores similares, excepto para lo siguiente:

- El campo *MDPRI* se puede establecer en PRQDEF (el gestor de colas lo cambiará a la prioridad predeterminada para la cola de inicio cuando se coloque el mensaje).
- El campo *MDRM* se puede establecer en blancos (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando coloque el mensaje).

- Los campos de contexto deben establecerse según corresponda para la aplicación.

Campos

La estructura MQTM contiene los campos siguientes; los campos se describen en **orden alfabético**:

TMAI (serie de caracteres de 256 bytes)

Identificador de aplicación.

Es una serie de caracteres que identifica la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ApplId** del objeto de proceso identificado por el campo *TMPN* ; consulte "Atributos para definiciones de proceso en IBM i" en la página 1432 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

El significado de *TMAI* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *TMAI* sea el nombre de un programa ejecutable.

La longitud de este campo la proporciona LNPROA. El valor inicial de este campo es de 256 caracteres en blanco.

TMAT (entero con signo de 10 dígitos)

Tipo de aplicación.

Esto identifica la naturaleza del programa que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ApplType** del objeto de proceso identificado por el campo *TMPN* ; consulte "Atributos para definiciones de proceso en IBM i" en la página 1432 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

TMAT puede tener uno de los siguientes valores estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a los valores del rango ATUFST a través de ATULST:

a la(s)CICS
Transacción CICS .

ATVSE
Transacción CICS/VSE .

AT400
IBM i .

ATUFST
Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST
Valor más alto para el tipo de aplicación definido por el usuario.

El valor inicial de este campo es 0.

TMED (serie de caracteres de 128 bytes)

Datos de entorno.

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **EnvData** del objeto de proceso identificado por el campo *TMPN* ; consulte "Atributos para definiciones de proceso en IBM i" en la página 1432 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

La longitud de este campo la proporciona LNPROE. El valor inicial de este campo es de 128 caracteres en blanco.

TMPN (serie de caracteres de 48 bytes)

Nombre del objeto de proceso.

Es el nombre del objeto de proceso del gestor de colas especificado para la cola desencadenada y puede ser utilizado por la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **ProcessName** de la cola identificada por el campo *TMQN* ; consulte "Atributos para colas" en la página 1402 para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo siempre se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona LNPRON. El valor inicial de este campo es de 48 caracteres en blanco.

TMQN (serie de caracteres de 48 bytes)

Nombre de la cola desencadenada.

Es el nombre de la cola para la que se ha producido un suceso desencadenante y lo utiliza la aplicación iniciada por la aplicación de supervisor desencadenante. El gestor de colas inicializa este campo con el valor del atributo **QName** de la cola desencadenada; consulte "Atributos para colas" en la página 1402 para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

TMSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

TMSIDV
Identificador de la estructura del mensaje desencadenante.

El valor inicial de este campo es TMSIDV.

TMTD (serie de caracteres de 64 bytes)

Datos del desencadenante.

Se trata de datos de formato libre para que los utilice la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **TriggerData** de la cola identificada por el campo *TMQN* ; consulte "Atributos para colas" en la página 1402 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

La longitud de este campo la proporciona LNTRGD. El valor inicial de este campo es de 64 caracteres en blanco.

TMUD (serie de caracteres de 128 bytes)

Datos de usuario.

Es una serie de caracteres que contiene información de usuario relevante para la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo **UserData** del objeto de proceso identificado por el campo *TMPN* ; consulte "Atributos para definiciones de proceso en IBM i" en la página 1432 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

La longitud de este campo la proporciona LNPROU. El valor inicial de este campo es de 128 caracteres en blanco.

TMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

TMVER1
Número de versión para la estructura del mensaje desencadenante.

La constante siguiente especifica el número de versión de la versión actual:

TMVERC
Versión actual de la estructura del mensaje desencadenante.

El valor inicial de este campo es TMVER1.

Valores iniciales

Tabla 737. Valores iniciales de campos en MQTM

Nombre de campo	Nombre de constante	Valor de constante
TMSID	TMSIDV	'TM¬¬'
TMVER	TMVER1	1
TMQN	Ninguna	Espacios en blanco
TMPN	Ninguna	Espacios en blanco
TMTD	Ninguna	Espacios en blanco
TMAT	Ninguna	0
TMAI	Ninguna	Espacios en blanco
TMED	Ninguna	Espacios en blanco
TMUD	Ninguna	Espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (Formato de 2 caracteres de mensaje desencadenante) en IBM i

Cuando una aplicación de supervisor desencadenante recupera un mensaje desencadenante (MQTM) de una cola de inicio, es posible que el supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación iniciada por el supervisor desencadenante.

Visión general

Finalidad: la información que puede necesitar la aplicación iniciada incluye *TC2QN*, *TC2TD*y *TC2UD*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada.

Esta estructura forma parte de IBM MQ Trigger Monitor Interface (TMI), que es una de las interfaces de infraestructura de IBM MQ .

Conjunto de caracteres y codificación: los datos de caracteres de MQTMC2 están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo del gestor de colas **CodedCharSetId** .

Uso: la estructura MQTMC2 es como el formato de la estructura MQTM. La diferencia es que los campos que no son de tipo carácter en MQTM se cambian en MQTMC2 por campos de tipo carácter de la misma longitud, y el nombre del gestor de colas se añade al final de la estructura.

- En IBM i, la aplicación de supervisor desencadenante proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.
- “Campos” en la página 1269
- “Valores iniciales” en la página 1270
- “Declaración RPG” en la página 1271

Campos

La estructura MQTMC2 contiene los campos siguientes; los campos se describen en **orden alfabético**:

TC2AI (serie de caracteres de 256 bytes)

Identificador de aplicación.

Consulte el campo *TMAI* en la estructura MQTM.

TC2AT (serie de caracteres de 4 bytes)

Tipo de aplicación.

Este campo siempre contiene espacios en blanco, cualquiera que sea el valor del campo *TMAT* en la estructura MQTM del mensaje desencadenante original.

TC2ED (serie de caracteres de 128 bytes)

Datos de entorno.

Consulte el campo *TMED* en la estructura MQTM.

TC2PN (serie de caracteres de 48 bytes)

Nombre del objeto de proceso.

Consulte el campo *TMPN* en la estructura MQTM.

TC2QMN (serie de caracteres de 48 bytes)

Nombre del gestor de colas.

Es el nombre del gestor de colas en el que se ha producido el suceso desencadenante.

TC2QN (serie de caracteres de 48 bytes)

Nombre de la cola desencadenada.

Consulte el campo *TMQN* en la estructura MQTM.

TC2SID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

TCSIDV
Identificador de la estructura del mensaje desencadenante (formato de caracteres).

TC2TD (serie de caracteres de 64 bytes)

Datos del desencadenante.

Consulte el campo *TMTD* en la estructura MQTM.

TC2UD (serie de caracteres de 128 bytes)

Datos de usuario.

Consulte el campo *TMUD* en la estructura MQTM.

TC2VER (serie de caracteres de 4 bytes)

Número de versión de la estructura.

El valor debe ser:

TCVER2
Estructura del mensaje desencadenante de la versión 2 (formato de caracteres).

La constante siguiente especifica el número de versión de la versión actual:

TCVERC
Versión actual de la estructura del mensaje desencadenante (formato de caracteres).

Valores iniciales

Tabla 738. Valores iniciales de los campos en MQTMC2

Nombre de campo	Nombre de constante	Valor de constante
TC2SID	TCSIDV	`'TMC¬'`
TC2VER	TCVER2	`'¬¬¬2'`
TC2QN	Ninguna	Espacios en blanco
TC2PN	Ninguna	Espacios en blanco
TC2TD	Ninguna	Espacios en blanco
TC2AT	Ninguna	Espacios en blanco
TC2AI	Ninguna	Espacios en blanco
TC2ED	Ninguna	Espacios en blanco
TC2UD	Ninguna	Espacios en blanco
TC2QMN	Ninguna	Espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración 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 (cabecera de información de trabajo) en IBM i

La estructura MQWIH describe la información que debe estar presente al principio de un mensaje que debe ser manejado por el gestor de carga de trabajo de z/OS .

Visión general

Nombre de formato: FMWIH.

Juego de caracteres y codificación: los campos de la estructura MQWIH están en el juego de caracteres y la codificación proporcionados por los campos *MDCSI* y *MDENC* de la estructura de cabecera que precede a MQWIH, o por los campos de la estructura MQMD si MQWIH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Uso: si el gestor de carga de trabajo de z/OS va a procesar un mensaje, el mensaje debe empezar con una estructura MQWIH.

- “Campos” en la página 1271
- “Valores iniciales” en la página 1273
- “Declaración RPG” en la página 1274

Campos

La estructura MQWIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

WICSI (entero con signo de 10 dígitos)

Identificador de juego de caracteres de los datos que siguen a MQWIH.

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQWIH; no se aplica a los datos de tipo carácter de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devolverá el valor CSINHT.

CSINHT no se puede utilizar si el valor del campo *MDPAT* en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

WIENC (entero con signo de 10 dígitos)

Codificación numérica de datos que sigue a MQWIH.

Especifica la codificación numérica de los datos que siguen a la estructura MQWIH; no se aplica a los datos numéricos de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

WIFLG (entero con signo de 10 dígitos)

Distintivos

El valor debe ser:

GANADERÍA
Sin distintivos.

El valor inicial de este campo es WINONE.

WIFMT (serie de caracteres de 8 bytes)

Nombre de formato de los datos que siguen a MQWIH.

Especifica el nombre de formato de los datos que siguen a la estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *MDFMT* en MQMD.

La longitud de este campo la proporciona LNFMT. El valor inicial de este campo es FMNONE.

WILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQWIH.

El valor debe ser:

WILEN1
Longitud de la estructura de cabecera de información de trabajo version-1 .

La constante siguiente especifica la longitud de la versión actual:

WILENC
Longitud de la versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es WILEN1.

WIRSV (serie de caracteres de 32 bytes)

Reservado.

Es un campo reservado; debe estar en blanco.

WISID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

WISIDV
Identificador de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es WISIDV.

WISNM (serie de caracteres de 32 bytes)

Nombre de servicio.

Es el nombre del servicio que va a procesar el mensaje.

La longitud de este campo la proporciona LNSVNM. El valor inicial de este campo es de 32 caracteres en blanco.

WISST (serie de caracteres de 8 bytes)

Nombre de paso de servicio.

Es el nombre del paso de *WISNM* con el que se relaciona el mensaje.

La longitud de este campo la proporciona LNSVST. El valor inicial de este campo es de 8 caracteres en blanco.

WITOK (serie de bits de 16 bytes)

Señal de mensaje.

Se trata de una señal de mensaje que identifica de forma exclusiva el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo se ignora. La longitud de este campo la proporciona LNMTOK. El valor inicial de este campo es MTKNON.

WIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

WIVER1
Estructura de cabecera de información de trabajo Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

WIVERC
Versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es WIVER1.

Valores iniciales

Tabla 739. Valores iniciales de campos en MQWIH

Nombre de campo	Nombre de constante	Valor de constante
WISID	WISIDV	'WIH¬'
WIVER	WIVER1	1
WILEN	WILEN1	120
WIENC	Ninguna	0
WICSI	CUNDF	0
WIFMT	FMNONE	Espacios en blanco
WIFLG	GANADERÍA	0
WISNM	Ninguna	Espacios en blanco
WISST	Ninguna	Espacios en blanco

Tabla 739. Valores iniciales de campos en MQWIH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
WITOK	MTKNON	Nulos
WIRSV	Ninguna	Espacios en blanco
Notas: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración 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 (cabecera de cola de transmisión) en IBM i

La estructura MQXQH describe la información que se añade como prefijo a los datos de mensaje de aplicación de los mensajes cuando están en colas de transmisión.

Visión general

Finalidad: una cola de transmisión es un tipo especial de cola local que contiene temporalmente mensajes destinados a colas remotas (es decir, destinados a colas que no pertenecen al gestor de colas local). Una cola de transmisión se indica mediante el atributo de cola **Usage** que tiene el valor USTRAN.

Nombre de formato: FMXQH.

Conjunto de caracteres y codificación: los datos de MQXQH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación de MQXQH deben establecerse en los campos *MDCSI* y *MDENC* en:

- El MQMD separado (si la estructura MQXQH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQXQH (todos los demás casos).

Uso: un mensaje que está en una cola de transmisión tiene *dos* descriptores de mensaje:

- Un descriptor de mensaje se almacena por separado de los datos de mensaje; esto se denomina *descriptor de mensaje separado*y lo genera el gestor de colas cuando el mensaje se coloca en la cola

de transmisión. Algunos de los campos del descriptor de mensaje separado se copian del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

El descriptor de mensaje separado es el que se devuelve a la aplicación en el parámetro **MSGDSC** de la llamada MQGET cuando se elimina el mensaje de la cola de transmisión.

- Un segundo descriptor de mensaje se almacena dentro de la estructura MQXQH como parte de los datos del mensaje; esto se denomina *descriptor de mensaje incorporado*y es una copia del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 (con variaciones menores).

  El descriptor de mensaje incorporado es siempre un MQMD version-1 . Si el mensaje colocado por la aplicación tiene valores no predeterminados para uno o más de los campos version-2 en MQMD, una estructura MQMDE sigue a la MQXQH y, a su vez, va seguida de los datos del mensaje de la aplicación (si los hay). El MQMDE es:

  – Generado por el gestor de colas (si la aplicación utiliza un MQMD version-2 para transferir el mensaje), o
  – Ya está presente al principio de los datos del mensaje de aplicación (si la aplicación utiliza un MQMD version-1 para transferir el mensaje).

  El descriptor de mensaje incorporado es el que se devuelve a la aplicación en el parámetro **MSGDSC** de la llamada MQGET cuando se elimina el mensaje de la cola de destino final.

- “Campos en el descriptor de mensaje separado” en la página 1275
- “Campos en el descriptor de mensaje incorporado” en la página 1276
- “Transferir mensajes a colas remotas” en la página 1277
- “Colocación de mensajes directamente en colas de transmisión” en la página 1277
- “Obtención de mensajes de colas de transmisión” en la página 1277
- “Campos” en la página 1277
- “Valores iniciales” en la página 1278
- “Declaración RPG” en la página 1279

Campos en el descriptor de mensaje separado

El gestor de colas establece los campos en el descriptor de mensaje separado tal como se muestra en la lista siguiente. Si el gestor de colas no da soporte al MQMD version-2 , se utiliza un MQMD version-1 sin pérdida de función.

Tabla 740. Campos en el descriptor de mensaje independiente y valores utilizados

Campo en MQMD separado	Valor utilizado
MDSID	MDSIDV
MDVER	MDVER2
MDREP	Se copia desde el descriptor de mensaje incorporado, pero con los bits identificados por ROAUXM establecidos en cero. (Esto impide que se genere un mensaje de informe COA o COD cuando se coloca un mensaje en o se elimina de una cola de transmisión.)
MDMT	Copiado del descriptor de mensaje incorporado.
MDEXP	Copiado del descriptor de mensaje incorporado.
MDFB	Copiado del descriptor de mensaje incorporado.
MDENC	ENNAT
MDCSI	Atributo **CodedCharSetId** del gestor de colas.
MDFMT	FMXQH

Tabla 740. Campos en el descriptor de mensaje independiente y valores utilizados (continuación)

Campo en MQMD separado	Valor utilizado
MDPRI	Copiado del descriptor de mensaje incorporado.
MDPER	Copiado del descriptor de mensaje incorporado.
MDMID	El gestor de colas genera un nuevo valor. Este identificador de mensaje es diferente del *MDMID* que el gestor de colas puede haber generado para el descriptor de mensaje incorporado (consulte la descripción anterior).
MDCID	*MDMID* del descriptor de mensaje incorporado.
MDBOC	0
MDRQ	Copiado del descriptor de mensaje incorporado.
MDRM	Copiado del descriptor de mensaje incorporado.
MDUID	Copiado del descriptor de mensaje incorporado.
MDACC	Copiado del descriptor de mensaje incorporado.
MDAID	Copiado del descriptor de mensaje incorporado.
MDPAT	ATQM
MDPAN	Primeros 28 bytes del nombre del gestor de colas.
MDPD	Fecha en la que se colocó el mensaje en la cola de transmisión.
MDPT	Hora en que se colocó el mensaje en la cola de transmisión.
MDAOD	Espacios en blanco
MDGID	GINONA
MDSEQ	1
MDOFF	0
MDMFL	MFNONE
MDOLN	OLUNDF

Campos en el descriptor de mensaje incorporado

Los campos del descriptor de mensaje incorporado tienen los mismos valores que los del parámetro **MSGDSC** de la llamada MQPUT o MQPUT1 , excepto los siguientes:

- El campo *MDVER* siempre tiene el valor MDVER1.
- Si el campo *MDPRI* tiene el valor PRQDEF, se sustituye por el valor del atributo **DefPriority** de la cola.
- Si el campo *MDPER* tiene el valor PEQDEF, se sustituye por el valor del atributo **DefPersistence** de la cola.
- Si el campo *MDMID* tiene el valor MINONE, o se ha especificado la opción PMNMID, o el mensaje es un mensaje de lista de distribución, *MDMID* se sustituye por un nuevo identificador de mensaje generado por el gestor de colas.

  Cuando un mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños colocados en diferentes colas de transmisión, el campo *MDMID* de cada uno de los nuevos descriptores de mensajes incorporados es el mismo que el del mensaje de lista de distribución original.
- Si se ha especificado la opción PMNCID, *MDCID* se sustituye por un nuevo identificador de correlación generado por el gestor de colas.
- Los campos de contexto se establecen tal como indican las opciones PM* especificadas en el parámetro **PMO** ; los campos de contexto son:

- *MDACC*
- *MDAID*
- *MDAOD*
- *MDPAN*
- *MDPAT*
- *MDPD*
- *MDPT*
- *MDUID*

- Los campos version-2 (si estaban presentes) se eliminan del MQMD y se mueven a una estructura MQMDE, si uno o varios de los campos version-2 tienen un valor no predeterminado.

Transferir mensajes a colas remotas

: Cuando una aplicación coloca un mensaje en una cola remota (especificando directamente el nombre de la cola remota o utilizando una definición local de la cola remota), el gestor de colas local:

- Crea una estructura MQXQH que contiene el descriptor de mensaje incorporado
- Añade un MQMDE si se necesita uno y no está ya presente
- Añade los datos de mensaje de aplicación
- Coloca el mensaje en una cola de transmisión adecuada

Colocación de mensajes directamente en colas de transmisión

También es posible que una aplicación coloque un mensaje directamente en una cola de transmisión. En este caso, la aplicación debe prefijar los datos del mensaje de aplicación con una estructura MQXQH e inicializar los campos con los valores adecuados. Además, el campo *MDFMT* del parámetro **MSGDSC** de la llamada MQPUT o MQPUT1 debe tener el valor FMXQH.

Los datos de tipo carácter de la estructura MQXQH creada por la aplicación deben estar en el juego de caracteres del gestor de colas local (definido por el atributo de gestor de colas **CodedCharSetId**) y los datos enteros deben estar en la codificación de máquina nativa. Además, los datos de tipo carácter de la estructura MQXQH deben rellenarse con espacios en blanco hasta la longitud definida del campo; los datos no deben finalizarse prematuramente utilizando un carácter nulo, porque el gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQXQH.

Sin embargo, tenga en cuenta que el gestor de colas no comprueba si existe una estructura MQXQH o que se han especificado valores válidos para los campos.

Obtención de mensajes de colas de transmisión

Las aplicaciones que obtienen mensajes de una cola de transmisión deben procesar la información en la estructura MQXQH de una forma adecuada. La presencia de la estructura MQXQH al principio de los datos del mensaje de aplicación se indica mediante el valor FMXQH que se devuelve en el campo *MDFMT* del parámetro **MSGDSC** de la llamada MQGET. Los valores devueltos en los campos *MDCSI* y *MDENC* en el parámetro **MSGDSC** , indican el juego de caracteres y la codificación de los datos de caracteres y enteros en la estructura MQXQH. El juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos *MDCSI* y *MDENC* en el descriptor de mensaje incorporado.

Campos

La estructura MQXQH contiene los campos siguientes; los campos se describen en **orden alfabético**:

XQMD (MQMD1)

Descriptor de mensaje original.

Este es el descriptor de mensaje incorporado y es una copia cercana del MQMD del descriptor de mensaje que se ha especificado como parámetro **MSGDSC** en la llamada MQPUT o MQPUT1 cuando el mensaje se colocó originalmente en la cola remota.

Nota: Se trata de un MQMD de version-1 .

Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQMD.

XQRQ (serie de caracteres de 48 bytes)

Nombre de la cola de destino.

Este es el nombre de la cola de mensajes que es el destino final aparente del mensaje (puede que no sea el destino final real si, por ejemplo, esta cola se define en *XQRQM* para que sea una definición local de otra cola remota).

Si el mensaje es un mensaje de lista de distribución (es decir, el campo *MDFMT* del descriptor de mensaje incorporado es FMDH), *XQRQ* está en blanco.

La longitud de este campo la proporciona LNQN. El valor inicial de este campo es de 48 caracteres en blanco.

XQRQM (serie de caracteres de 48 bytes)

Nombre del gestor de colas de destino.

Es el nombre del gestor de colas o del grupo de compartición de colas que es propietario de la cola que es el destino aparente final del mensaje.

Si el mensaje es un mensaje de lista de distribución, *XQRQM* está en blanco.

La longitud de este campo la proporciona LNQMN. El valor inicial de este campo es de 48 caracteres en blanco.

XQSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

XQSIDV
Identificador de la estructura de cabecera de cola de transmisión.

El valor inicial de este campo es XQSIDV.

XQVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

XQVER1
Número de versión para la estructura de cabecera de cola de transmisión.

La constante siguiente especifica el número de versión de la versión actual:

XQVERC
Versión actual de la estructura de cabecera de cola de transmisión.

El valor inicial de este campo es XQVER1.

Valores iniciales

Tabla 741. Valores iniciales de los campos en MQXQH

Nombre de campo	Nombre de constante	Valor de constante
XQSID	XQSIDV	'XQH¬'
XQVER	XQVER1	1

Tabla 741. Valores iniciales de los campos en MQXQH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
XQRQ	Ninguna	Espacios en blanco
XQRQM	Ninguna	Espacios en blanco
XQMD	Los mismos nombres y valores que MQMD; consulte Tabla 709 en la página 1177	-
Notas: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID                  1      4    INZ('XQH ')
D* Structure version number
D  XQVER                  5      8I 0 INZ(1)
D* Name of destination queue
D  XQRQ                   9     56    INZ
D* Name of destination queue manager
D  XQRQM                 57    104    INZ
D* Original message descriptor
D  XQ1SID               105    108    INZ('MD  ')
D  XQ1VER               109    112I 0 INZ(1)
D  XQ1REP               113    116I 0 INZ(0)
D  XQ1MT                117    120I 0 INZ(8)
D  XQ1EXP               121    124I 0 INZ(-1)
D  XQ1FB                125    128I 0 INZ(0)
D  XQ1ENC               129    132I 0 INZ(273)
D  XQ1CSI               133    136I 0 INZ(0)
D  XQ1FMT               137    144    INZ('        ')
D  XQ1PRI               145    148I 0 INZ(-1)
D  XQ1PER               149    152I 0 INZ(2)
D  XQ1MID               153    176    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1CID               177    200    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1BOC               201    204I 0 INZ(0)
D  XQ1RQ                205    252    INZ
D  XQ1RM                253    300    INZ
D  XQ1UID               301    312    INZ
D  XQ1ACC               313    344    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     00000000000000000000-
D                                     000000')
D  XQ1AID               345    376    INZ
D  XQ1PAT               377    380I 0 INZ(0)
D  XQ1PAN               381    408    INZ
D  XQ1PD                409    416    INZ
D  XQ1PT                417    424    INZ
D  XQ1AOD               425    428    INZ
```

IBM i Llamadas de función en IBM i

Utilice esta información para obtener información sobre las llamadas de función disponibles en la programación de IBM i .

Convenciones utilizadas en las descripciones de llamada en IBM i

Para cada llamada, esta colección de temas proporciona una descripción de los parámetros y el uso de la llamada. Esto va seguido de invocaciones típicas de la llamada, y declaraciones típicas de sus parámetros, en el lenguaje de programación RPG.

Importante: Al codificar llamadas de API de IBM MQ , debe asegurarse de que se proporcionan todos los parámetros relevantes (tal como se describe en las secciones siguientes). No hacerlo puede producir resultados imprevisibles.

La descripción de cada llamada contiene las secciones siguientes:

Nombre de llamada
El nombre de la llamada, seguido de una breve descripción de la finalidad de la llamada.

Parámetros
Para cada parámetro, el nombre va seguido de su tipo de datos entre paréntesis () y su dirección; por ejemplo:

CMPCOD (entero decimal de 9 dígitos)-salida

Hay más información sobre los tipos de datos de estructura en "Tipos de datos elementales" en la página 1017.

La dirección del parámetro puede ser:

Entrada
Usted (el programador) debe proporcionar este parámetro.

Salida
La llamada devuelve este parámetro.

Entrada/salida
Debe proporcionar este parámetro, pero la llamada lo modifica.

También hay una breve descripción de la finalidad del parámetro, junto con una lista de los valores que puede tomar el parámetro.

Los dos últimos parámetros de cada llamada son un código de terminación y un código de razón. El código de terminación indica si la llamada se ha completado correctamente, parcialmente o no. En el código de razón se proporciona más información sobre el éxito parcial o el fracaso de la llamada.

Notas de uso
Información adicional sobre la llamada, describiendo cómo utilizarla y cualquier restricción sobre su uso.

Invocación de RPG
Invocación típica de la llamada, y declaración de sus parámetros, en RPG.

Otras convenciones notariales son:

Constantes
Los nombres de constantes se muestran en mayúsculas; por ejemplo, OOOUT.

Matrices
En algunas llamadas, los parámetros son matrices de series de caracteres con un tamaño que no es fijo. En las descripciones de estos parámetros, una *n* en minúsculas representa una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya *n* por el valor numérico que necesita.

IBM i MQBACK (Retrotraer cambios) en IBM i

La llamada MQBACK indica al gestor de colas que todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización deben restituirse. Los mensajes colocados como parte de una unidad de trabajo se suprimen; los mensajes recuperados como parte de una unidad de trabajo se restablecen en la cola.

- Esta llamada está soportada en los entornos siguientes:
  - AIX
  - IBM i
  - Solaris
  - Windows
- “Sintaxis” en la página 1281
- “Notas de uso” en la página 1281
- “Parámetros” en la página 1282
- “Declaración RPG” en la página 1283

Sintaxis

MQBACK *(Hconn, CompCode, Reason)*

Notas de uso

Tenga en cuenta estas notas de uso al utilizar MQBACK.

1. Esta llamada sólo se puede utilizar cuando el propio gestor de colas coordina la unidad de trabajo. Se trata de una unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una retrotracción implícita causada por la terminación anómala de la aplicación.
   - En IBM i, esta llamada se puede utilizar para unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en “MQDISC (Desconectar gestor de colas) en IBM i” en la página 1321 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *MDGID*, *MDSEQ*, *MDOFF*y *MDMFL* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   El gestor de colas mantiene *tres* conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:
   - La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).

   Si la aplicación coloca u obtiene los mensajes como parte de una unidad de trabajo y, a continuación, la aplicación decide restituir la unidad de trabajo, la información de grupo y segmento se restaura al valor que tenía anteriormente:

- La información asociada con la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT satisfactoria para ese manejador de cola en la unidad de trabajo actual.
- La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

Las colas actualizadas por la aplicación después de que se haya iniciado la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen restaurada la información de grupo y segmento si se restituye la unidad de trabajo.

La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo. El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema. Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en "MQPMO (opciones de transferencia de mensajes) en IBM i" en la página 1200y la opción GMLOGO descrita en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098.

Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:

1. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Esto significa que todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro **HCONN** descrito en "MQCONN (Conectar gestor de colas) en IBM i" en la página 1307 para obtener información sobre el ámbito de los manejadores de conexión.
2. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
3. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede hacer que las colas se llenen con mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQBACK tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *COMCOD*.

Si *COMCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *COMCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2123
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQBACK(HCONN : COMCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Iniciar unidad de trabajo) en IBM i

La llamada MQBEGIN inicia una unidad de trabajo coordinada por el gestor de colas y que puede implicar a gestores de recursos externos.

- Esta llamada está soportada en los entornos siguientes:
  - AIX AIX
  - IBM i IBM i

– Solaris Solaris

– Windows Windows

- “Sintaxis” en la página 1284
- “Notas de uso” en la página 1284
- “Parámetros” en la página 1285
- “Declaración RPG” en la página 1286

Sintaxis

MQBEGIN *(HCONN, BEGOP, CMPCOD, REASON)*

Notas de uso

1. La llamada MQBEGIN se puede utilizar para iniciar una unidad de trabajo coordinada por el gestor de colas y que puede implicar cambios en los recursos propiedad de otros gestores de recursos. El gestor de colas da soporte a tres tipos de unidad de trabajo:

   Unidad de trabajo local coordinada por el gestor de colas
   Es una unidad de trabajo en la que el gestor de colas es el único gestor de recursos que participa y, por lo tanto, el gestor de colas actúa como coordinador de unidad de trabajo.

   - Para iniciar este tipo de unidad de trabajo, se debe especificar la opción PMSYP o GMSYP en la primera llamada MQPUT, MQPUT1o MQGET en la unidad de trabajo.

     No es necesario que la aplicación emita la llamada MQBEGIN para iniciar la unidad de trabajo, pero si se utiliza MQBEGIN, la llamada se completa con CCWARN y el código de razón RC2121.
   - Para confirmar o restituir este tipo de unidad de trabajo, debe utilizarse la llamada MQCMIT o MQBACK.

   Unidad de trabajo global coordinada por el gestor de colas
   Se trata de una unidad de trabajo en la que el gestor de colas actúa como coordinador de unidad de trabajo, tanto para IBM MQ recursos *como para* para recursos que pertenecen a otros gestores de recursos. Estos gestores de recursos cooperan con el gestor de colas para asegurarse de que todos los cambios en los recursos de la unidad de trabajo se confirman o se restituyen juntos.

   - Para iniciar este tipo de unidad de trabajo, debe utilizarse la llamada MQBEGIN.
   - Para confirmar o restituir este tipo de unidad de trabajo, se deben utilizar las llamadas MQCMIT y MQBACK.

   Unidad de trabajo global coordinada externamente
   Se trata de una unidad de trabajo en la que el gestor de colas es un participante, pero el gestor de colas no actúa como coordinador de unidad de trabajo. En su lugar, hay un coordinador de unidad de trabajo externo con el que el gestor de colas coopera.

   - Para iniciar este tipo de unidad de trabajo, se debe utilizar la llamada pertinente proporcionada por el coordinador externo de la unidad de trabajo.

     Si se utiliza la llamada MQBEGIN para intentar iniciar la unidad de trabajo, la llamada falla con el código de razón RC2012.
   - Para confirmar o restituir este tipo de unidad de trabajo, se deben utilizar las llamadas de confirmación y de devolución proporcionadas por el coordinador de unidad de trabajo externo.

     Si se utiliza la llamada MQCMIT o MQBACK para intentar confirmar o restituir la unidad de trabajo, la llamada falla con el código de razón RC2012.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en “MQDISC (Desconectar gestor de colas) en IBM i” en la página 1321 para obtener más detalles.

3. Una aplicación sólo puede participar en una unidad de trabajo a la vez. La llamada MQBEGIN falla con el código de razón RC2128 si ya existe una unidad de trabajo para la aplicación, independientemente del tipo de unidad de trabajo que sea.
4. La llamada MQBEGIN no es válida en un entorno de cliente IBM MQ . Un intento de utilizar la llamada falla con el código de razón RC2012.
5. Cuando el gestor de colas actúa como coordinador de unidad de trabajo para unidades de trabajo globales, los gestores de recursos que pueden participar en la unidad de trabajo se definen en el archivo de configuración del gestor de colas.
6. En IBM i, los tres tipos de unidad de trabajo están soportados de la siguiente manera:
   - **Las unidades de trabajo locales coordinadas por el gestor de colas** sólo se pueden utilizar cuando no existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no se debe haber emitido para el trabajo.
   - **Las unidades de trabajo globales coordinadas por el gestor de colas** no están soportadas.
   - Las **unidades de trabajo globales coordinadas externamente** solo se pueden utilizar cuando existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** debe haberse emitido para el trabajo. Si esto se ha hecho, las operaciones IBM i COMMIT y ROLLBACK se aplican a los recursos IBM MQ , así como a los recursos que pertenecen a otros gestores de recursos participantes.

Parámetros

La llamada MQBEGIN tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

BEGOP (MQBO)-entrada/salida

Opciones que controlan la acción de MQBEGIN.

Consulte “MQBO (opciones de inicio) en IBM i” en la página 1039 para obtener los detalles.

Si no se necesitan opciones, los programas escritos en C o S/390 assembler pueden especificar una dirección de parámetro nula, en lugar de especificar la dirección de una estructura MQBO.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2121
(2121, X'849 ') No se ha registrado ningún gestor de recursos participante.

RC2122
(2122, X'84A') El gestor de recursos participante no está disponible.

Si *CMPCOD* es CCFAIL:

RC2134
(2134, X'856 ') Estructura de opciones de inicio no válida.

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2012
(2012, X'7DC') La llamada no es válida en el entorno.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2128
(2128, X'850 ') Unidad de trabajo ya iniciada.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQBEGIN(HCONN : BEGOP : CMPCOD :
C                                       REASON)
```

La definición de prototipo para la llamada es:

```
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 (Convertir almacenamiento intermedio en descriptor de mensaje) en IBM i

La llamada de función MQBUFMH convierte un almacenamiento intermedio en un manejador de mensajes y es el inverso de la llamada MQMHBUF.

Esta llamada toma un descriptor de mensaje y las propiedades MQRFH2 del almacenamiento intermedio y las hace disponibles a través de un descriptor de mensaje. Las propiedades MQRFH2 de los datos del mensaje se eliminan, opcionalmente. Los campos *Encoding*, *CodedCharSetId*y *Format* del descriptor de mensaje se actualizan, si es necesario, para describir correctamente el contenido del almacenamiento intermedio después de que se hayan eliminado las propiedades.

- "Sintaxis" en la página 1287
- "Notas de uso" en la página 1287
- "Parámetros" en la página 1287
- "Declaración RPG" en la página 1289

Sintaxis

MQBUFMH *(Hconn, Hmsg, BufMsgHOpts, MsgDesc, Buffer, BufferLength, DataLength, CompCode, Reason)*

Notas de uso

Las llamadas MQBUFMH no se pueden interceptar mediante salidas de API: un almacenamiento intermedio se convierte en un manejador de mensajes en el espacio de aplicación; la llamada no llega al gestor de colas.

Parámetros

La llamada MQBUFMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra convirtiendo un almacenamiento intermedio en un descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este descriptor de contexto es el descriptor de contexto de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

BMHOPT (MQBMHO)-entrada

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios.

Consulte "MQBMHO (opciones de almacenamiento intermedio a manejador de mensajes) en IBM i" en la página 1038 para obtener los detalles.

MSGDSC (MQMD)-entrada/salida

La estructura *MSGDSC* contiene las propiedades del descriptor de mensaje y describe el contenido del área de almacenamiento intermedio.

En la salida de la llamada, las propiedades se eliminan opcionalmente del área de almacenamiento intermedio y, en este caso, el descriptor de mensaje se actualiza para describir correctamente el área de almacenamiento intermedio.

Los datos de esta estructura deben estar en el juego de caracteres y la codificación de la aplicación.

BUFLEN (entero con signo de 10 dígitos)-entrada

BUFLEN es la longitud del área de almacenamiento intermedio, en bytes.

Un *BUFLEN* de cero bytes es válido e indica que el área de almacenamiento intermedio no contiene datos.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada/salida

BUFFER define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BUFFER* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro **MSGDSC** en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BUFLEN** es cero, no se hace referencia a *BUFFER* . En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DATLEN (entero con signo de 10 dígitos)-salida

DATLEN es la longitud, en bytes, del almacenamiento intermedio que puede tener las propiedades eliminadas.

CMPCOD (entero con signo de 10 dígitos)-salida

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2489
(2489, X'09B9') La estructura de opciones de almacenamiento intermedio a manejador de mensajes no es válida.

RC2004
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

RC2005
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2009
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2460
(2460, X'099C') Descriptor de mensaje no válido.

RC2026
(2026, X'07EA') Descriptor de mensaje no válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2334
(2334, X'091E') La estructura MQRFH2 no es válida.

RC2421
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQBUFMH(HCONN : HMSG : BMHOPT :
                                        MSGDSC : BUFLEN : BUFFER :
                                        DATLEN : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Gestionar devolución de llamada) en IBM i

La llamada MQCB vuelve a registrar una devolución de llamada para el descriptor de objeto especificado y controla la activación y los cambios en la devolución de llamada.

Una devolución de llamada es un fragmento de código (especificado como el nombre de una función que se puede enlazar dinámicamente o como un puntero de función) al que llama IBM MQ cuando se producen determinados sucesos.

Para utilizar MQCB y MQCTL en un cliente V7 debe estar conectado a un servidor V7 y el parámetro **SHARECNV** del canal debe tener un valor distinto de cero.

Para obtener información sobre las unidades de trabajo globales, consulte: Unidades de trabajo globales.

Los tipos de devolución de llamada que se pueden definir son:

Consumidor de mensajes

Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto.

Sólo se puede registrar una función de devolución de llamada para cada descriptor de contexto de objeto. Si se va a leer una sola cola con varios criterios de selección, la cola se debe abrir varias veces y se debe registrar una función de consumidor en cada descriptor de contexto.

Manejador de sucesos

Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de devolución de llamada.

Se llama a la función cuando se produce una condición de suceso, por ejemplo, un gestor de colas o una conexión que se detiene o se desactiva temporalmente.

La función no se invoca para condiciones específicas de un único consumidor de mensajes, por ejemplo, RC2016; sin embargo, se llama si una función de devolución de llamada no finaliza normalmente.

- "Sintaxis" en la página 1290
- "Notas de uso para MQCB" en la página 1290
- "Parámetros para MQCB" en la página 1291
- "Declaración RPG" en la página 1298

Sintaxis

MQCB *(HCONN, OPERATN, HOBJ, CBDSC, MSGDSC, GMO, CMPCOD, REASON)*

Notas de uso para MQCB

1. MQCB se utiliza para definir la acción que se va a invocar para cada mensaje, que coincide con los criterios especificados, disponibles en la cola. Cuando se procesa la acción, el mensaje se elimina de la cola y se pasa al consumidor de mensajes definido, o se proporciona una señal de mensaje, que se utiliza para recuperar el mensaje.
2. MQCB se puede utilizar para definir rutinas de devolución de llamada antes de iniciar el consumo con MQCTL o se puede utilizar desde dentro de una rutina de devolución de llamada.
3. Para utilizar MQCB desde fuera de una rutina de devolución de llamada, primero debe suspender el consumo de mensajes utilizando MQCTL y reanudar el consumo posteriormente.

Secuencia de devolución de llamada de consumidor de mensajes

Puede configurar un consumidor para invocar la devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor esté registrado por primera vez,
- cuando se inicia la conexión,
- cuando se detiene la conexión y
- cuando se anula el registro del consumidor, ya sea explícita o implícitamente mediante un MQCLOSE.

Tabla 742. Definiciones de verbo MQCTL

Verbo	Significado
MQCTL (START)	Llamada MQCTL utilizando la operación CTLSR
MQCTL (STOP)	Llamada MQCTL utilizando la operación CTLSP
MQCTL (ESPERAR)	Llamada MQCTL utilizando la operación CTLSW

Permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación de consumidor son las siguientes:

REGISTER
Es siempre el primer tipo de invocación de la devolución de llamada.

Siempre se llama en la misma hebra que la llamada MQCB (CBREG).

START
Siempre se llama de forma síncrona con el verbo MQCTL (START).

- Todas las devoluciones de llamada START se completan antes de que se devuelva el verbo MQCTL (START).

Está en la misma hebra que la entrega de mensajes si se solicita CTLTHR.

La llamada con inicio no se garantiza si, por ejemplo, una devolución de llamada anterior emite MQCTL (STOP) durante MQCTL (START).

STOP
No se entregan más mensajes o sucesos después de esta llamada hasta que se reinicia la conexión.

Se garantiza un STOP si la aplicación se ha llamado anteriormente para START, o un mensaje, o un suceso.

DEREGISTER
Es siempre el último tipo de invocación de la devolución de llamada.

Asegúrese de que la aplicación realiza la inicialización y limpieza basadas en hebras en las devoluciones de llamada START y STOP. Puede realizar una inicialización y limpieza no basadas en hebras con devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida y la disponibilidad del hilo aparte de lo que se indica. Por ejemplo, no confíe en que una hebra permanezca activa más allá de la última llamada a DEREGISTER. De forma similar, cuando haya elegido no utilizar CTLTHR, no presuponga que la hebra existe siempre que se inicie la conexión.

Si la aplicación tiene requisitos específicos para las características de hebra, siempre puede crear una hebra en consecuencia y, a continuación, utilizar MQCTL (WAIT). Este paso *dona* la hebra a IBM MQ para la entrega de mensajes asíncronos.

Uso de conexión de consumidor de mensajes

Normalmente, cuando una aplicación emite otra llamada MQI mientras hay una pendiente, la llamada falla con el código de razón RC2219.

Sin embargo, hay casos especiales en los que la aplicación debe emitir una llamada MQI adicional antes de que se haya completado la llamada anterior. Por ejemplo, el consumidor se puede invocar durante una llamada MQCB con CBRE.

En tal caso, cuando como resultado de la aplicación que emite un verbo MQCB o MQCTL, se devuelve la llamada a la aplicación, la aplicación puede emitir una llamada MQI adicional. Esta instancia significa que puede emitir, por ejemplo, una llamada MQOPEN, en la función de consumidor cuando se llama con un tipo CBCCALLT de CBCTRC. Se permite cualquier llamada MQI, excepto MQDISC.

Parámetros para MQCB

La llamada MQCB tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Gestionar función de devolución de llamada-parámetro HCONN.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

OPERATN (entero con signo de 10 dígitos)-entrada

Gestionar función de devolución de llamada-Parámetro OPERATN.

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una de las opciones siguientes; si se necesita más de una opción, los valores pueden añadirse (no añadir la misma constante más de una vez) o combinarse utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bits).

Las combinaciones que no son válidas se anotan; todas las demás combinaciones son válidas.

CBREG

Defina la función de devolución de llamada para el descriptor de objeto especificado. Esta operación define la función que se va a llamar y los criterios de selección que se van a utilizar.

Si ya se ha definido una función de devolución de llamada para el descriptor de contexto de objeto, se sustituye la definición. Si se detecta un error al sustituir la devolución de llamada, se anula el registro de la función.

Si una devolución de llamada se registra en la misma función de devolución de llamada en la que se ha anulado el registro anteriormente, se trata como una operación de sustitución; no se invoca ninguna llamada inicial o final.

Puede utilizar CBREG con CTLSU o CTLRE.

CUNR

Detiene el consumo de mensajes para el descriptor de contexto de objeto y elimina el descriptor de contexto de los elegibles para una devolución de llamada.

Una devolución de llamada se anula automáticamente si se cierra el descriptor de contexto asociado.

Si se llama a CBUNR desde dentro de un consumidor, y la devolución de llamada tiene definida una llamada de detención, se invoca tras la devolución del consumidor.

Si esta operación se emite para un *Hobj* sin ningún consumidor registrado, la llamada se devuelve con RC2448.

CTLSU

Suspende el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

Mientras está suspendida, la función de consumidor continúa obteniendo las devoluciones de llamada de tipo de control.

CTLRE

Reanude el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

CBDSC (MQCBD)-entrada

Gestionar función de devolución de llamada-Parámetro CBDSC.

Esta es una estructura que identifica la función de devolución de llamada que está registrando la aplicación y las opciones utilizadas al registrarla.

Consulte “MQCBD-Descriptor de devolución de llamada” en la página 287 para obtener detalles de la estructura.

El descriptor de devolución de llamada sólo es necesario para la opción CBREG; si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

HOBJ (entero con signo de 10 dígitos)-entrada

Gestionar función de devolución de llamada-Parámetro HOBJ.

Este descriptor de contexto representa el acceso que se ha establecido al objeto desde el que se va a consumir un mensaje. Es un descriptor de contexto que se ha devuelto desde una llamada MQOPEN o MQSUB anterior (en el parámetro **HOBJ**).

HOBJ no es necesario al definir una rutina de manejador de sucesos (CBTEH) y debe especificarse como HONONE.

Si esta *Hobj* se ha devuelto desde una llamada MQOPEN, la cola debe haberse abierto con una o más de las opciones siguientes:

- OOINPS
- OOINPX
- OOINPQ
- OOBRW

MSGDSC (MQMD)-entrada

Gestionar función de devolución de llamada -parámetro MSGDSC.

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado.

El parámetro **MsgDesc** define los atributos de los mensajes que necesita el consumidor y la versión del MQMD que se va a pasar al consumidor de mensajes.

MsgId, *CorrelId*, *GroupId*, *MsgSeqNumber*y *Offset* en MQMD se utilizan para la selección de mensajes, en función de las opciones especificadas en el parámetro **GetMsgOpts** .

Encoding y *CodedCharSetId* se utilizan para la conversión de mensajes si especifica la opción GMCONV.

Consulte MQMD para obtener más detalles.

MsgDesc sólo se utiliza para CBREG y, si necesita valores distintos del valor predeterminado para cualquier campo. *MsgDesc* no se utiliza para un manejador de sucesos.

Si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

Tenga en cuenta que si se registran varios consumidores en la misma cola con selectores solapados, el consumidor elegido para cada mensaje no está definido.

GMO (MQGMO)-entrada

Gestionar función de devolución de llamada-Parámetro GMO.

Opciones que controlan cómo el consumidor de mensajes obtiene los mensajes.

Todas las opciones tienen el significado que se describe en “MQGMO (Opciones de obtención de mensajes) en IBM i” en la página 1098, cuando se utilizan en una llamada MQGET, excepto:

GMSSIG
Esta opción no está permitida.

GMBRWF, GMBRWN, GMMBH, GMMBC
El orden de los mensajes entregados a un consumidor de navegación viene determinado por las combinaciones de estas opciones. Las combinaciones significativas son:

GMBRWF
El primer mensaje de la cola se entrega repetidamente al consumidor. Esto es útil cuando el consumidor consume de forma destructiva el mensaje en la devolución de llamada. Utilice esta opción con cuidado.

GMBRWN
Al consumidor se le asigna cada mensaje de la cola, desde la posición actual del cursor hasta que se alcanza el final de la cola.

GMBRWF + GMBRWN
El cursor se restablece al inicio de la cola. A continuación, se proporciona al consumidor cada mensaje hasta que el cursor llega al final de la cola.

GMBRWF + GMMBH o GMMBC
A partir del principio de la cola, al consumidor se le proporciona el primer mensaje no marcado en la cola, que a continuación se marca para este consumidor. Esta combinación garantiza que el consumidor pueda recibir nuevos mensajes añadidos detrás del punto de cursor actual.

GMBRWN + GMMBH o GMMBC
A partir de la posición del cursor, el consumidor recibe el siguiente mensaje no marcado en la cola, que a continuación se marca para este consumidor. Utilice esta combinación con cuidado porque los mensajes se pueden añadir a la cola detrás de la posición actual del cursor.

GMBRWF + GMBRWN + GMMBH o GMMBC
Esta combinación no está permitida, si se utiliza, la llamada devuelve RC2046.

GMNWT, GMWT y GMWI
Estas opciones controlan cómo se invoca al consumidor.

GMNWT
Nunca se llama al consumidor con RC2033. El consumidor sólo se invoca para mensajes y sucesos

GMWT con GMWI cero
El código RC2033 sólo se pasa al consumidor cuando no hay mensajes y

- el consumidor se ha iniciado
- el consumidor ha recibido al menos un mensaje desde el último código de razón sin mensajes.

Esto impide que el consumidor sondee en un bucle ocupado cuando se especifica un intervalo de espera cero.

GMWT y una GMWI positiva
El usuario se invoca después del intervalo de espera especificado con el código de razón RC2033. Esta llamada se realiza independientemente de si se ha entregado algún mensaje al consumidor. Esto permite al usuario realizar un proceso de latido o de tipo de proceso por lotes.

GMWT y GMWI de WIULIM
Especifica una espera infinita antes de devolver RC2033. Nunca se llama al consumidor con RC2033.

GMO sólo se utiliza para CBREG y, si necesita valores distintos del valor predeterminado para cualquier campo. *GMO* no se utiliza para un manejador de sucesos.

Si las opciones no son necesarias, la dirección de parámetro pasada puede ser nula.

Si se proporciona un descriptor de contexto de propiedades de mensaje en la estructura MQGMO, se proporciona una copia en la estructura MQGMO que se pasa a la devolución de llamada del consumidor. Al volver de la llamada MQCB, la aplicación puede suprimir el descriptor de contexto de propiedades de mensaje.

CMPCOD (entero con signo de 10 dígitos)-salida

Gestionar función de devolución de llamada-Parámetro CMPCOD.

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Gestionar función de devolución de llamada-parámetro REASON.

Los siguientes códigos de razón son los códigos que el gestor de colas puede devolver para el parámetro **REASON** .

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es CCFAIL:

RC2204
(2204, X'89C') El adaptador no está disponible.

RC2133
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2374
(2374, X'946') La salida de la API ha fallado.

RC2183
(2183, X'887') No se ha podido cargar la salida de la API.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2005
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2487
(2487, X'9B7') Campo de tipo de devolución de llamada incorrecto.

RC2448
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada.

RC2486
(2486, X'9B6') Se debe especificar *CallbackFunction* o *CallbackName* , pero no ambos.

RC2483
(2483, X'9B3') Campo de tipo de devolución de llamada incorrecto.

RC2484
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

RC2140
(2140, X'85C') Solicitud de espera rechazada por CICS.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2217
(2217, X'8A9') No tiene autorización para la conexión.

RC2202
(2202, X'89A') Conexión en fase de inmovilización.

RC2203
(2203, X'89B') La conexión está concluyendo.

RC2207
(2207, X'89F') Error de identificador de correlación.

RC2010
(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2016
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2351
(2351, X'92F') Conflicto de unidades de trabajo global.

RC2186
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2353
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2259
(2259, X'8D3') La especificación de examinar es incoherente.

RC2245
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2352
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

RC2247
(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2485
(2485, X'9B4') Campo *MaxMsgLength* incorrecto.

RC2026
(2026, X'7EA') El descriptor de mensaje no es válido.

RC2497
(2497, X'9C1') No se ha podido encontrar el punto de entrada de función especificado en el módulo.

RC2496
(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto; no de 32 bits, 64 bits o una biblioteca de enlace dinámico válida.

RC2495
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

RC2250
(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2331
(2331, X'91B') El uso del símbolo de mensaje no es válido.

RC2033
(2033, X'7F1') No hay ningún mensaje disponible.

RC2034
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

RC2036
(2036, X'7F4') La cola no se ha abierto para examen.

RC2037
(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2206
(2206, X'89E') Código de operación incorrecto en llamada de API.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2193
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2394
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2069
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2109
(2109, X'83D') El programa de salida ha suprimido la llamada.

RC2024
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2072
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2354
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

RC2355
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

RC2255
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2298
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCB(HCONN : OPERATN : CBDSC :
                                HOBJ : MSGDSC : GMO :
                                DATLEN : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Cerrar objeto) en IBM i

La llamada MQCLOSE renuncia al acceso a un objeto y es la inversa de la llamada MQOPEN.

- “Sintaxis” en la página 1298
- “Notas de uso” en la página 1298
- “Parámetros” en la página 1299
- “Declaración RPG” en la página 1304

Sintaxis

MQCLOSE *(HCONN, HOBJ, OPTS, CMPCOD, REASON)*

Notas de uso

1. Cuando una aplicación emite la llamada MQDISC, o finaliza de forma normal o anómala, los objetos abiertos por la aplicación y que siguen abiertos se cierran automáticamente con la opción CONONE.
2. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola*:
   - Si las operaciones en la cola se realizan como parte de una unidad de trabajo, la cola se puede cerrar antes o después de que se produzca el punto de sincronización sin afectar al resultado del punto de sincronización.
   - Si la cola se ha abierto con la opción OOBRW, el cursor para examinar se destruye. Si la cola se vuelve a abrir posteriormente con la opción OOBRW, se crea un nuevo cursor para examinar (consulte la opción OOBRW descrita en MQOPEN).
   - Si un mensaje está bloqueado actualmente para este descriptor de contexto en el momento de la llamada MQCLOSE, el bloqueo se libera (consulte la opción GMLK descrita en “MQGMO (Opciones de obtención de mensajes) en IBM i” en la página 1098).

3. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola dinámica* (permanente o temporal):
   - Para una cola dinámica, se pueden especificar las opciones CODEL o COPURG independientemente de las opciones especificadas en la llamada MQOPEN correspondiente.
   - Cuando se suprime una cola dinámica, se cancelan todas las llamadas MQGET con la opción GMWT que están pendientes en la cola y se devuelve el código de razón RC2052 . Consulte la opción GMWT descrita en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098.

     Después de que se haya suprimido una cola dinámica, cualquier llamada (distinta de MQCLOSE) que intente hacer referencia a la cola utilizando un descriptor de contexto *HOBJ* adquirido anteriormente falla con el código de razón RC2052.

     Tenga en cuenta que aunque las aplicaciones no pueden acceder a una cola suprimida, la cola no se elimina del sistema y los recursos asociados no se liberan, hasta que se hayan cerrado todos los descriptores de contexto que hacen referencia a la cola y todas las unidades de trabajo que afectan a la cola se hayan confirmado o restituido.
   - Cuando se suprime una cola dinámica permanente, si el descriptor de contexto *HOBJ* especificado en la llamada MQCLOSE no es el que ha devuelto la llamada MQOPEN que ha creado la cola, se comprueba que el identificador de usuario que se ha utilizado para validar la llamada MQOPEN esté autorizado para suprimir la cola. Si se ha especificado la opción OOALTU en la llamada MQOPEN, el identificador de usuario seleccionado es *ODAU*.

     Esta comprobación no se realiza si:
     - El descriptor de contexto especificado es el devuelto por la llamada MQOPEN que ha creado la cola.
     - La cola que se está suprimiendo es una cola dinámica temporal.
   - Cuando se cierra una cola dinámica temporal, si el descriptor de contexto *HOBJ* especificado en la llamada MQCLOSE es el que ha devuelto la llamada MQOPEN que ha creado la cola, se suprime la cola. Esto ocurre independientemente de las opciones de cierre especificadas en la llamada MQCLOSE. Si hay mensajes en la cola, se descartan; no se generan mensajes de informe.

     Si hay unidades de trabajo no confirmadas que afectan a la cola, la cola y sus mensajes se siguen suprimiendo, pero esto no hace que fallen las unidades de trabajo. Sin embargo, como se ha descrito anteriormente, los recursos asociados con las unidades de trabajo no se liberan hasta que se haya confirmado o restituido cada una de las unidades de trabajo.
4. Los puntos siguientes se aplican si el objeto que se está cerrando es una *lista de distribución*:
   - La única opción de cierre válida para una lista de distribución es CONONE; la llamada falla con el código de razón RC2046 o RC2045 si se especifica alguna otra opción.
   - Cuando se cierra una lista de distribución, los códigos de terminación y los códigos de razón individuales no se devuelven para las colas de la lista; sólo los parámetros **CMPCOD** y **REASON** de la llamada están disponibles para fines de diagnóstico.

     Si se produce una anomalía al cerrar una de las colas, el gestor de colas continúa el proceso e intenta cerrar las colas restantes de la lista de distribución. A continuación, los parámetros **CMPCOD** y **REASON** de la llamada se establecen para devolver información que describe la anomalía. Por lo tanto, es posible que el código de terminación sea CCFAIL, aunque la mayoría de las colas se hayan cerrado correctamente. La cola que ha encontrado el error no está identificada.

     Si hay una anomalía en más de una cola, no se define qué anomalía se notifica en los parámetros **CMPCOD** y **REASON** .

Parámetros

La llamada MQCLOSE tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada/salida

El manejador del objeto.

Este descriptor de contexto representa el objeto que se está cerrando. El objeto puede ser de cualquier tipo. El valor de *HOBJ* ha sido devuelto por una llamada MQOPEN anterior.

Al finalizar correctamente la llamada, el gestor de colas establece este parámetro en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

HOUNUH
Descriptor de objeto inutilizable.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQCLOSE.

El parámetro **OPTS** controla cómo se cierra el objeto. Sólo las colas dinámicas permanentes y las suscripciones se pueden cerrar de más de una forma. Las colas dinámicas permanentes se pueden retener o suprimir; son colas con un atributo **DefinitionType** que tiene el valor QDPERM (consulte el atributo **DefinitionType** descrito en "Atributos para colas" en la página 1402). Las opciones de cierre se resumen en una tabla más adelante en este tema.

Las suscripciones duraderas se pueden conservar o eliminar; estas se crean utilizando la llamada MQSUB con la opción SODUR.

Al cerrar el descriptor de contexto en un destino gestionado (es decir, el parámetro **Hobj** devuelto en una llamada MQSUB que ha utilizado la opción SOMAN), el gestor de colas limpiará las publicaciones no recuperadas cuando también se haya eliminado la suscripción asociada. Esto se realiza utilizando la opción CORMSB en el parámetro **Hsub** devuelto en una llamada MQSUB. Tenga en cuenta que CORMSB es el comportamiento predeterminado en MQCLOSE para una suscripción no duradera.

Al cerrar un descriptor de contexto en un destino no gestionado, es responsable de limpiar la cola donde se envían las publicaciones. Se recomienda cerrar primero la suscripción utilizando CORMSB y, a continuación, procesar los mensajes fuera de la cola hasta que no quede ninguno.

Se debe especificar uno (y sólo uno) de los siguientes:

Opciones de cierre de cola dinámica

Estas opciones controlan cómo se cierran las colas dinámicas permanentes:

MODELO
Suprima la cola.

La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay mensajes en la cola y no hay solicitudes de obtención o colocación no confirmadas pendientes para la cola (para la tarea actual o para cualquier otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *HOBJ*. En este caso, se depuran todos los mensajes de la cola.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón RC2045y el objeto no se suprime.

COPURG
Suprima la cola, depurando los mensajes que haya en ella.

La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay solicitudes get o put no confirmadas pendientes para la cola (ni para la tarea actual ni para ninguna otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *HOBJ*.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón RC2045y el objeto no se suprime.

La tabla siguiente muestra qué opciones de cierre son válidas y si el objeto se conserva o se suprime.

Tabla 743. Opciones de cierre válidas para utilizar con objetos retenidos o suprimidos

Tipo de objeto o cola	CONONA	MODELO	COPURG
Objeto que no es una cola	Retenido	No válido	No válido
Cola predefinida	Retenido	No válido	No válido
cola dinámica permanente	Retenido	Suprimido si está vacío y no hay actualizaciones pendientes	Mensajes suprimidos; cola suprimida si no hay actualizaciones pendientes
Cola dinámica temporal (llamada emitida por el creador de la cola)	Suprimida	Suprimida	Suprimida
Cola dinámica temporal (llamada no emitida por el creador de la cola)	Retenido	No válido	No válido
Lista de distribución	Retenido	No válido	No válido
Destino de suscripción gestionada	Retenido	No válido	No válido
Lista de distribución (se ha eliminado la suscripción)	Mensajes suprimidos; cola suprimida	No válido	No válido

Opciones de cierre de suscripción

Estas opciones controlan si se eliminan las suscripciones duraderas cuando se cierra el descriptor de contexto y si se limpian las publicaciones que todavía están a la espera de ser leídas por la aplicación. Estas opciones sólo son válidas para su uso con un descriptor de objeto devuelto en el parámetro **HSUB** de una llamada MQSUB.

COKPSB
El descriptor de contexto de la suscripción se cierra, pero la suscripción realizada se mantiene. Las publicaciones se seguirán enviando al destino especificado en la suscripción. Esta opción sólo es válida si la suscripción se ha realizado con la opción SODUR. COKPSB es el valor predeterminado si la suscripción es duradera

CORMSB
La suscripción se elimina y se cierra el descriptor de contexto de la suscripción.

El parámetro **Hobj** de la llamada MQSUB no se invalida mediante el cierre del parámetro **Hsub** y puede seguir utilizándose para MQGET o MQCB para recibir las publicaciones restantes. Cuando el parámetro **Hobj** de la llamada MQSUB también se cierra, si era un destino gestionado, se eliminarán las publicaciones no recuperadas.

CORMSB es el valor predeterminado si la suscripción no es duradera.

Estas opciones de cierre de suscripción se resumen en las tablas siguientes:

Para cerrar un descriptor de contexto de suscripción duradera pero dejar la suscripción alrededor, utilice las siguientes opciones de cierre de suscripción:

Tabla 744. Opciones de tarea para cerrar un descriptor de contexto de suscripción duradera y dejar la suscripción alrededor	
Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	COKPSB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto con SOMAN	COKPSB
Eliminar publicaciones en un descriptor de contexto con SOMAN	Acción no permitida

Para anular la suscripción, ya sea cerrando un descriptor de contexto de suscripción duradera y cancelándola o cerrando un descriptor de contexto de suscripción no duradera, utilice las siguientes opciones de cierre de suscripción:

Tabla 745. Opciones de tarea para anular la suscripción	
Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	CORMSB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto con SOMAN	CORMSB
Eliminar publicaciones en un descriptor de contexto con SOMAN	COPGSB

Opciones de lectura anticipada

Las opciones siguientes controlan lo que sucede con los mensajes no persistentes que se han enviado al cliente antes de que una aplicación los solicitara y que todavía no han sido consumidos por la aplicación. Estos mensajes se almacenan en el almacenamiento intermedio de lectura anticipada del cliente a la espera de ser solicitados por la aplicación y se pueden descartar o consumir de la cola antes de que se complete MQCLOSE.

CIMM
El objeto se cierra inmediatamente y los mensajes que se han enviado al cliente antes de que una aplicación los solicitara se descartan y no están disponibles para que los consuma ninguna aplicación. Éste es el valor predeterminado.

COQSC
Se realiza una solicitud para cerrar el objeto, pero si alguno de los mensajes que se han enviado al cliente antes de que una aplicación los solicitara, sigue residiendo en el almacenamiento intermedio de lectura anticipada del cliente, la llamada MQCLOSE devolverá un código de aviso de RC2458y el descriptor de contexto del objeto seguirá siendo válido.

A continuación, la aplicación puede seguir utilizando el descriptor de objeto para recuperar mensajes hasta que no haya más disponibles y, a continuación, vuelva a cerrar el objeto. No se enviarán más mensajes al cliente antes de que se solicite una aplicación, la lectura anticipada está ahora desactivada.

Se recomienda a las aplicaciones que utilicen COQSC en lugar de intentar llegar a un punto en el que no haya más mensajes en el almacenamiento intermedio de lectura anticipada del cliente,

ya que podría llegar un mensaje entre la última llamada MQGET y el siguiente MQCLOSE que se descartaría si se utilizara COIMM.

Si se emite un MQCLOSE con COQSC desde una función de devolución de llamada asíncrona, se aplica el mismo comportamiento de los mensajes de lectura anticipada. Si se devuelve el código de aviso RC2458 , se llamará a la función de devolución de llamada al menos una vez más. Cuando el último mensaje restante que se ha leído con anticipación se ha pasado a la función de devolución de llamada, el campo CBCFLG se establece en CBCFBE.

Opción predeterminada

Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

CONONA
No es necesario ningún proceso de cierre opcional.

Debe especificarse para:

- Objetos que no son colas
- Colas predefinidas
- Colas dinámicas temporales (pero sólo en los casos en los que *HOBJ* no es el manejador devuelto por la llamada MQOPEN que ha creado la cola).
- Listas de distribución

En todos los casos anteriores, el objeto se conserva y no se suprime.

Si se especifica esta opción para una cola dinámica temporal:

- La cola se suprime, si la ha creado la llamada MQOPEN que ha devuelto *HOBJ* ; los mensajes que están en la cola se depuran.
- En todos los demás casos, la cola (y los mensajes que contiene) se retienen.

Si se especifica esta opción para una cola dinámica permanente, la cola se conserva y no se suprime.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2241
(2241, X'8C1') El grupo de mensajes no está completo.

RC2242
(2242, X'8C2') El mensaje lógico no está completo.

Si *CMPCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2035
(2035, X'7F3') No autorizado para el acceso.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2045
(2045, X'7FD') Opción no válida para el tipo de objeto.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2055
(2055, X'807 ') La cola contiene uno o más mensajes o solicitudes put u get no confirmadas.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2063
(2063, X'80F') Se ha producido un error de seguridad.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCLOSE(HCONN : HOBJ : OPTS :
C                                       CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Confirmar cambios) en IBM i

La llamada MQCMIT indica al gestor de colas que la aplicación ha alcanzado un punto de sincronización y que todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización se deben hacer permanentes. Los mensajes colocados como parte de una unidad de trabajo se ponen a disposición de otras aplicaciones; los mensajes recuperados como parte de una unidad de trabajo se suprimen.

- "Sintaxis" en la página 1305
- "Notas de uso" en la página 1305
- "Parámetros" en la página 1306
- "Declaración RPG" en la página 1307

Sintaxis

MQCMIT *(HCONN, COMCOD, REASON)*

Notas de uso

Tenga en cuenta estas notas de uso cuando utilice MQCMIT.

1. Esta llamada sólo se puede utilizar cuando el propio gestor de colas coordina la unidad de trabajo. Se trata de una unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de confirmación adecuada en lugar de MQCMIT. El entorno también puede dar soporte a una confirmación implícita causada por la terminación normal de la aplicación.
   - En IBM i, esta llamada se puede utilizar para unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC (Desconectar gestor de colas) en IBM i" en la página 1321 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *MDGID*, *MDSEQ*, *MDOFF*y *MDMFL* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   Cuando se confirma una unidad de trabajo, el gestor de colas conserva la información de grupo y segmento, y la aplicación puede continuar colocando u obteniendo mensajes en el grupo de mensajes o mensaje lógico actual.

   La retención de la información de grupo y segmento cuando se confirma una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo. El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema. Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en "MQPMO (opciones de transferencia de mensajes) en IBM i" en la página 1200y la opción GMLOGO descrita en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098.

Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:

1. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Esto significa que todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro **HCONN** descrito en MQCONN para obtener información sobre el ámbito de los manejadores de conexión.
2. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
3. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o de devolución, puede hacer que las colas se llenen con mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQCMIT tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

COMCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *COMCOD*.

Si *COMCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *COMCOD* es CCWARN:

RC2003
(2003, X'7D3') Unidad de trabajo restituida.

RC2124
(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *COMCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2123
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCMIT(HCONN : COMCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Conectar gestor de colas) en IBM i

La llamada MQCONN conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas de cola de mensajes posteriores.

- Las aplicaciones deben utilizar la llamada MQCONN o MQCONNX para conectarse al gestor de colas y la llamada MQDISC para desconectarse del gestor de colas.

En IBM MQ for Windows, UNIXy IBM i, cada hebra de una aplicación puede conectarse a distintos gestores de colas. En otros sistemas, todas las conexiones simultáneas dentro de un proceso deben estar en el mismo gestor de colas.

- “Sintaxis” en la página 1307
- “Notas de uso” en la página 1308
- “Parámetros” en la página 1308
- “Declaración RPG” en la página 1311

Sintaxis

MQCONN *(QMNAME, HCONN, CMPCOD, REASON)*

Notas de uso

1. El gestor de colas con el que se realiza la conexión utilizando la llamada MQCONN se denomina *gestor de colas local*.
2. Las colas que son propiedad del gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas compartidas que son propiedad del grupo de compartición de colas al que pertenece el gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas que son propiedad de gestores de colas remotos aparecen como colas remotas. Es posible colocar mensajes en estas colas, pero no es posible obtener mensajes de estas colas.
3. Si el gestor de colas falla mientras una aplicación se está ejecutando, la aplicación debe volver a emitir la llamada MQCONN para obtener un nuevo descriptor de conexión que utilizar en las llamadas IBM MQ posteriores. La aplicación puede emitir la llamada MQCONN periódicamente hasta que la llamada sea satisfactoria.

   Si una aplicación no está segura de si está conectada al gestor de colas, la aplicación puede emitir de forma segura una llamada MQCONN para obtener un descriptor de conexión. Si la aplicación ya está conectada, el descriptor de contexto devuelto es el mismo que el devuelto por la llamada MQCONN anterior, pero con el código de terminación CCWARN y el código de razón RC2002.
4. Cuando la aplicación haya terminado de utilizar las llamadas IBM MQ , la aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
5. En IBM i, los programas que finalizan de forma anómala no se desconectan automáticamente del gestor de colas. Por lo tanto, las aplicaciones deben escribirse para permitir la posibilidad de que la llamada MQCONN o MQCONNX devuelva el código de terminación CCWARN y el código de razón RC2002. El descriptor de conexión devuelto en esta situación se puede utilizar como normal.

Parámetros

La llamada MQCONN tiene los parámetros siguientes:

QMNAME (serie de caracteres de 48 bytes)-entrada

Nombre del gestor de colas.

Es el nombre del gestor de colas al que la aplicación desea conectarse. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Se puede utilizar un carácter nulo para indicar el final de datos significativos en el nombre; el nulo y cualquier carácter que lo siga se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben especificarse entre comillas cuando se especifican en mandatos. Estas comillas no se deben especificar en el parámetro **QMNAME** .

Si el nombre consta por completo de espacios en blanco, se utiliza el nombre del gestor de colas *por omisión* .

El nombre especificado para *QMNAME* debe ser el nombre de un gestor de colas *conectable* .

Grupos de compartición de colas:En sistemas donde existen varios gestores de colas y están configurados para formar un grupo de compartición de colas, el nombre del grupo de compartición de colas se puede especificar para *QMNAME* en lugar del nombre de un gestor de colas. Esto permite a la

aplicación conectarse a *cualquier* gestor de colas que esté disponible en el grupo de compartición de colas. El sistema también se puede configurar para que un *QMNAME* en blanco provoque la conexión con el grupo de compartición de colas en lugar de con el gestor de colas predeterminado.

Si *QMNAME* especifica el nombre del grupo de compartición de colas, pero también hay un gestor de colas con ese nombre en el sistema, se establece una conexión con el último en lugar del primero. Sólo si la conexión falla, se intenta la conexión con uno de los gestores de colas del grupo de compartición de colas.

Si la conexión es satisfactoria, el descriptor de contexto devuelto por la llamada MQCONN o MQCONNX se puede utilizar para acceder a *todos* los recursos (compartidos y no compartidos) que pertenecen al gestor de colas concreto con el que se ha realizado la conexión. El acceso a estos recursos está sujeto a los controles de autorización típicos.

Si la aplicación emite dos llamadas MQCONN o MQCONNX para establecer conexiones simultáneas, y una o ambas llamadas especifica el nombre del grupo de compartición de colas, la segunda llamada puede devolver el código de terminación CCWARN y el código de razón RC2002. Esto ocurre cuando la segunda llamada se conecta al mismo gestor de colas que la primera llamada.

Los grupos de compartición de colas solo se admiten en z/OS. La conexión a un grupo de compartición de colas sólo está soportada en los entornos por lotes, por lotes RRS y TSO.

IBM MQ aplicaciones cliente: Para aplicaciones IBM MQ MQI client , se intenta una conexión para cada definición de canal de conexión de cliente con el nombre de gestor de colas especificado, hasta que una sea satisfactoria. Sin embargo, el gestor de colas debe tener el mismo nombre que el nombre especificado. Si se especifica un nombre todo en blanco, se intenta cada canal de conexión de cliente con un nombre de gestor de colas todo en blanco hasta que uno sea satisfactorio; en este caso, no hay ninguna comprobación con respecto al nombre real del gestor de colas.

Grupos de gestores de colas de cliente IBM MQ: si el nombre especificado empieza con un asterisco (*), el gestor de colas real con el que se realiza la conexión puede tener un nombre distinto del especificado por la aplicación. El nombre especificado (sin el asterisco) define un *grupo* de gestores de colas que son elegibles para la conexión. La implementación selecciona uno del grupo intentando cada uno a su vez, en orden alfabético, hasta que se encuentra uno al que se puede realizar una conexión. Si ninguno de los gestores de colas del grupo está disponible para la conexión, la llamada falla. Cada gestor de colas se intenta una sola vez. Si se especifica un asterisco solo para el nombre, se utiliza un grupo de gestores de colas predeterminado definido por la implementación.

Los grupos de gestores de colas sólo están soportados para las aplicaciones que se ejecutan en un entorno de cliente MQ; la llamada falla si una aplicación no cliente especifica un nombre de gestor de colas que empieza con un asterisco. Un grupo se define proporcionando varias definiciones de canal de conexión de cliente con el mismo nombre de gestor de colas (el nombre especificado sin el asterisco), para comunicarse con cada uno de los gestores de colas del grupo. El grupo predeterminado se define proporcionando una o más definiciones de canal de conexión de cliente, cada una con un nombre de gestor de colas en blanco (por lo tanto, especificar un nombre en blanco tiene el mismo efecto que especificar un solo asterisco para el nombre de una aplicación cliente).

Después de conectarse a un gestor de colas de un grupo, una aplicación puede especificar espacios en blanco de la forma habitual en los campos de nombre de gestor de colas en los descriptores de mensaje y objeto para indicar el nombre del gestor de colas al que se ha conectado realmente la aplicación (el *gestor de colas local*). Si la aplicación necesita conocer este nombre, se puede emitir la llamada MQINQ para consultar el atributo del gestor de colas **QMgrName** .

El prefijo de un asterisco al nombre de conexión implica que la aplicación no depende de la conexión con un gestor de colas determinado del grupo. Las aplicaciones adecuadas serían:

- Aplicaciones que colocan mensajes pero no obtienen mensajes.
- Aplicaciones que colocan mensajes de solicitud y, a continuación, obtienen los mensajes de respuesta de una cola *dinámica temporal* .

Las aplicaciones no adecuadas serían aquellas que necesitan obtener mensajes de una cola determinada en un gestor de colas determinado; dichas aplicaciones no deben añadir un asterisco al nombre.

Tenga en cuenta que si se especifica un asterisco, la longitud máxima del resto del nombre es de 47 caracteres.

La longitud de este parámetro la proporciona LNQMN.

HCONN (entero con signo de 10 dígitos)-salida

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. Debe especificarse en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto está restringido a la unidad más pequeña de proceso paralelo soportado por la plataforma en la que se ejecuta la aplicación; el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada MQCONN.

- En IBM i, el ámbito del descriptor de contexto es el trabajo que emite la llamada.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2002
(2002, X'7D2') La aplicación ya está conectada.

Si *CMPCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2267
(2267, X'8DB') No se puede cargar la salida de carga de trabajo de clúster.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2035
(2035, X'7F3') No autorizado para el acceso.

RC2137
(2137, X'859 ') El objeto no se ha abierto correctamente.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2063
(2063, X'80F') Se ha producido un error de seguridad.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                      REASON)
```

La definición de prototipo para la llamada es:

```
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 (Conectar gestor de colas (ampliado)) en IBM i

La llamada MQCONNX conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas IBM MQ posteriores.

La llamada MQCONNX es como la llamada MQCONN, excepto que MQCONNX permite especificar opciones para controlar la forma en que funciona la llamada.

En IBM MQ for Windows, UNIXy IBM i, cada hebra de una aplicación puede conectarse a distintos gestores de colas. En otros sistemas, todas las conexiones simultáneas dentro de un proceso deben estar en el mismo gestor de colas.

- “Sintaxis” en la página 1311
- “Parámetros” en la página 1311
- “Declaración RPG” en la página 1312

Sintaxis

MQCONNX *(QMNAME, CNOPT, HCONN, CMPCOD, REASON)*

Parámetros

La llamada MQCONNX tiene los parámetros siguientes:

QMNAME (serie de caracteres de 48 bytes)-entrada

Nombre del gestor de colas.

Consulte el parámetro **QMNAME** descrito en “MQCONN (Conectar gestor de colas) en IBM i” en la página 1307 para obtener más detalles.

CNOPT (MQCNO)-entrada/salida

Opciones que controlan la acción de MQCONNX.

Consulte “MQCNO (opciones de conexión) en IBM i” en la página 1069 para obtener los detalles.

HCONN (entero con signo de 10 dígitos)-salida

Descriptor de conexión.

Consulte el parámetro **HCONN** descrito en “MQCONN (Conectar gestor de colas) en IBM i” en la página 1307 para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Consulte el parámetro **CMPCOD** descrito en “MQCONN (Conectar gestor de colas) en IBM i” en la página 1307 para obtener más detalles.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Consulte el parámetro **REASON** descrito en “MQCONN (Conectar gestor de colas) en IBM i” en la página 1307 para obtener detalles de los posibles códigos de razón.

La llamada MQCONNX puede devolver los siguientes códigos de razón adicionales:

Si *CMPCOD* es CCFAIL:

RC2278
(2278, X'8E6') Los campos de conexión de cliente no son válidos.

RC2139
(2139, X'85B') La estructura de opciones de conexión no es válida.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                      REASON)
```

La definición de prototipo para la llamada es:

```
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 (Crear manejador de mensajes) en IBM i

La llamada MQCRTMH devuelve un descriptor de mensaje.

Una aplicación puede utilizarla en llamadas de cola de mensajes posteriores:

- Utilice la llamada MQSETMP para establecer una propiedad del manejador de mensajes.
- Utilice la llamada MQINQMP para consultar el valor de una propiedad del manejador de mensajes.
- Utilice la llamada MQDLTMP para suprimir una propiedad del manejador de mensajes.

El descriptor de mensaje se puede utilizar en las llamadas MQPUT y MQPUT1 para asociar las propiedades del descriptor de mensaje con las propiedades del mensaje que se está colocando. De forma similar, al especificar un descriptor de mensaje en la llamada MQGET, se puede acceder a las propiedades del mensaje que se está recuperando utilizando el descriptor de mensaje cuando se completa la llamada MQGET.

Utilice MQDLTMH para suprimir el descriptor de mensaje.

- "Sintaxis" en la página 1313
- "Parámetros" en la página 1313
- "Declaración RPG" en la página 1315

Sintaxis

MQCRTMH *(Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason)*

Parámetros

La llamada MQCRTMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior. Si la conexión con el gestor de colas deja de ser válida y no hay ninguna llamada IBM MQ operativa en el manejador de mensajes, se llama implícitamente a MQDLTMH para suprimir el mensaje.

De forma alternativa, puede especificar el valor siguiente:

HCUNAS
El descriptor de conexión no representa una conexión con ningún gestor de colas en particular.

Cuando se utiliza este valor, el descriptor de mensaje debe suprimirse con una llamada explícita a MQDLTMH para liberar cualquier almacenamiento asignado a él; IBM MQ nunca suprime implícitamente el descriptor de mensaje.

Debe haber al menos una conexión válida con un gestor de colas establecido en la hebra que crea el manejador de mensajes; de lo contrario, la llamada falla con RC2018.

CRTOPT (MQCMHO)-entrada

Las opciones que controlan la acción de MQCRTMH. Consulte MQCMHO para obtener detalles.

HMSG (entero con signo de 20 dígitos)-salida

En la salida, se devuelve un descriptor de mensaje que se puede utilizar para establecer, consultar y suprimir propiedades del descriptor de mensaje. Inicialmente, el manejador de mensajes no contiene propiedades.

Un descriptor de mensaje también tiene un descriptor de mensaje asociado. Inicialmente, este descriptor de mensaje contiene los valores predeterminados. Los valores de los campos de descriptor de mensaje asociados se pueden establecer y consultar utilizando las llamadas MQSETMP y MQINQMP. La llamada MQDLTMP restablece un campo del descriptor de mensaje a su valor predeterminado.

Si se especifica el parámetro *HCONN* como el valor HCUNAS, el descriptor de mensaje devuelto se puede utilizar en llamadas MQGET, MQPUT o MQPUT1 con cualquier conexión dentro de la unidad de proceso, pero sólo puede estar en uso por una llamada IBM MQ cada vez. Si el descriptor de contexto está en uso cuando una segunda llamada IBM MQ intenta utilizar el mismo descriptor de contexto de mensaje, la segunda llamada IBM MQ falla con el código de razón RC2499.

Si el parámetro *HCONN* no es HCUNAS, el descriptor de mensaje devuelto sólo se puede utilizar en la conexión especificada.

Se debe utilizar el mismo valor de parámetro *HCONN* en las llamadas MQI posteriores donde se utiliza este descriptor de mensaje:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

El descriptor de mensaje devuelto deja de ser válido cuando se emite la llamada MQDLTMH para el descriptor de mensaje, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. MQDLTMH se llama implícitamente si se proporciona una conexión específica cuando se crea el descriptor de mensaje y la conexión con el gestor de colas deja de ser válida, por ejemplo, si se llama a MQDBC.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2461
(2461, X'099D') Crear estructura de opciones de manejador de mensajes no válida.

RC2273
(2273, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2017
(2017, X'07E1') No hay más manejadores disponibles.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2460
(2460, X'099C') El puntero de manejador de mensajes no es válido.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Consulte "Códigos de retorno para IBM i (ILE RPG)" en la página 1462 para obtener más detalles.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCRTMH(HCONN : CRTOPT : HMSG :
                                        CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (devolución de llamada de control) en IBM i

La llamada MQCTL realiza acciones de control en los manejadores de objetos abiertos para una conexión.

- "Sintaxis" en la página 1315
- "Notas de uso" en la página 1315
- "Parámetros" en la página 1315
- "Declaración RPG" en la página 1320

Sintaxis

MQCTL *(Hconn, Operation, ControlOpts, CompCode, Reason)*

Notas de uso

1. Las rutinas de devolución de llamada deben comprobar las respuestas de todos los servicios que invocan y, si la rutina detecta una condición que no se puede resolver, debe emitir un mandato MQCB (CBREG) para evitar llamadas repetidas a la rutina de devolución de llamada.

Parámetros

La llamada MQCTL tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

OPERATN (entero con signo de 10 dígitos)-entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una, y sólo una, de las opciones siguientes:

CTLSR

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Las devoluciones de llamada se ejecutan en una hebra iniciada por el sistema, que es diferente de cualquiera de las hebras de aplicación.

Esta operación proporciona el control del descriptor de conexión proporcionado al sistema. Las únicas llamadas MQI que puede emitir una hebra que no sea la hebra de consumidor son:

- MQCTL con operación CTLSP
- MQCTL con operación CTLSU
- MQDISC-Realiza MQCTL con la operación CTLSP antes de desconectar el HConn.

Se devuelve RC2500 si se emite una llamada de API IBM MQ mientras se inicia el descriptor de conexión y la llamada no se origina en una función de consumidor de mensajes.

Si una conexión falla, esto detiene la conversación tan pronto como sea posible. Por lo tanto, es posible que se emita una llamada de API de IBM MQ en la hebra principal para recibir el código de retorno RC2500 durante un tiempo, seguido del código de retorno RC2009 cuando la conexión vuelve al estado detenido.

Esto se puede emitir en una función de consumidor. Para la misma conexión que la rutina de devolución de llamada, su única finalidad es cancelar una operación CTLSP emitida anteriormente.

Esta opción no está soportada si la aplicación está enlazada con una biblioteca IBM MQ sin hebras.

CTLSW

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Los consumidores de mensajes se ejecutan en la misma hebra y el control no se devuelve al interlocutor de MQCTL hasta que:

- Liberado por el uso de las operaciones MQCTL CTLSP o CTLSU, o
- Todas las rutinas de consumidor se han desregistrado o suspendido.

Si se anula el registro o se suspenden todos los consumidores, se emite una operación CTLSP implícita.

Esta opción no se puede utilizar desde una rutina de devolución de llamada, ya sea para el descriptor de conexión actual o cualquier otro descriptor de conexión. Si se intenta la llamada, se devuelve con RC2012.

Si, en algún momento durante una operación CTLSW no hay ningún consumidor registrado, no suspendido, la llamada falla con un código de razón de RC2446.

Si, durante una operación CTLSW, la conexión se suspende, la llamada MQCTL devuelve un código de razón de aviso de RC2521; la conexión permanece 'iniciada'.

La aplicación puede elegir emitir CTLSP o CTLRE. En este caso, la operación CTLRE se bloquea.

Esta opción no está soportada en un cliente de una sola hebra.

CTLSP
Detenga el consumo de mensajes y espere a que todos los consumidores completen sus operaciones antes de que se complete esta opción. Esta operación libera el descriptor de conexión.

Si se emite desde dentro de una rutina de devolución de llamada, esta opción no entra en vigor hasta que se cierra la rutina. No se llama a más rutinas de consumidor de mensajes después de que se hayan completado las rutinas de consumidor para los mensajes que ya se han leído, y después de que se hayan realizado llamadas de detención (si se han solicitado) a rutinas de devolución de llamada.

Si se emite fuera de una rutina de devolución de llamada, el control no vuelve al llamante hasta que se hayan completado las rutinas del consumidor para los mensajes ya leídos y después de que se hayan realizado las llamadas de detención (si se han solicitado) a las devoluciones de llamada. Sin embargo, las devoluciones de llamada permanecen registradas.

Esta función no tiene ningún efecto en los mensajes de lectura anticipada. Debe asegurarse de que los consumidores ejecuten MQCLOSE (COQSC), desde dentro de la función de devolución de llamada, para determinar si hay más mensajes disponibles para entregar.

CTLSU
Pausar el consumo de mensajes. Esta operación libera el descriptor de conexión.

Esto no afecta a la lectura anticipada de los mensajes para la aplicación. Si tiene previsto dejar de consumir mensajes durante un periodo largo, considere la posibilidad de cerrar la cola y volver a abrirla cuando el consumo deba continuar.

Si se emite desde dentro de una rutina de devolución de llamada, no entra en vigor hasta que se cierra la rutina. No se llamarán más rutinas de consumidor de mensajes después de las salidas de rutina actuales.

Si se emite fuera de una devolución de llamada, el control no vuelve al interlocutor hasta que se haya completado la rutina de consumidor actual y no se llame a más.

CTLRE
Reanude el consumo de mensajes.

Esta opción se emite normalmente desde la hebra de aplicación principal, pero también se puede utilizar desde una rutina de devolución de llamada para cancelar una solicitud de suspensión anterior emitida en la misma rutina.

Si se utiliza CTLRE para reanudar un CTLSW, la operación se bloquea.

PCTLOP (MQCTLO)-entrada

Opciones que controlan la acción de MQCTL

Consulte MQCTLO para obtener detalles de la estructura.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Los siguientes códigos de razón son los que el gestor de colas puede devolver para el parámetro **Reason** .

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2133
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

RC2204
(2204, X'89C') El adaptador no está disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2374
(2374, X'946') La salida de la API ha fallado.

RC2183
(2183, X'887') No se ha podido cargar la salida de la API.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2005
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2487
(2487, X'9B7') No se puede llamar a la rutina de devolución de llamada

RC2448
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada

RC2486
(2486, X'9B6') Se ha especificado CallbackFunction y CallbackName en una llamada CBREG, o se ha especificado uno de CallbackFunction o CallbackName pero no coincide con la función de devolución de llamada registrada actualmente.

RC2483
(2483, X'9B3') Campo de tipo CallBackincorrecto.

RC2219
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2444
(2444, X'98C') El bloque de opciones es incorrecto.

RC2484
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

RC2140
(2140, X'85C') Solicitud de espera rechazada por CICS.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2217
(2217, X'8A9') No tiene autorización para la conexión.

RC2202
(2202, X'89A') Conexión en fase de inmovilización.

RC2203
(2203, X'89B') La conexión está concluyendo.

RC2207
(2207, X'89F') Error de identificador de correlación.

RC2016
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2351
(2351, X'92F') Conflicto de unidades de trabajo global.

RC2186
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2353
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2259
(2259, X'8D3') La especificación de examinar es incoherente.

RC2245
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2352
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

RC2247
(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2485
(2485, X'9B5') Campo de longitud MaxMsgincorrecto

RC2026
(2026, X'7EA') El descriptor de mensaje no es válido.

RC2497
(2497, X'9C1') No se ha encontrado el punto de entrada de función especificado en el módulo.

RC2496
(2496, X'9C0') Se ha encontrado el módulo pero es del tipo incorrecto (32 bits o 64 bits) o no es una dll válida.

RC2495
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

RC2206
(2206, X'89E') Error de identificador de mensaje.

RC2250
(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2331
(2331, X'91B') El uso del símbolo de mensaje no es válido.

RC2036
(2036, X'7F4') La cola no se ha abierto para examen.

RC2037
(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2488
(2488, X'9B8') Código de operación incorrecto en llamada de API

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2193
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2394
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2069
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2109
(2109, X'83D') El programa de salida ha suprimido la llamada.

RC2072
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2354
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

RC2355
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

RC2255
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2298
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCTL(HCONN : OPERATN : PCTLOP :
                                      CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Desconectar gestor de colas) en IBM i

La llamada MQDISC interrumpe la conexión entre el gestor de colas y el programa de aplicación, y es la inversa de la llamada MQCONN o MQCONNX.

- “Sintaxis” en la página 1321
- “Notas de uso” en la página 1321
- “Parámetros” en la página 1321
- “Declaración RPG” en la página 1322

Sintaxis

MQDISC *(HCONN, CMPCOD, REASON)*

Notas de uso

1. Si se emite una llamada MQDISC cuando la aplicación todavía tiene objetos abiertos, el gestor de colas cierra esos objetos, con las opciones de cierre establecidas en CONONE.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de cómo finalice la aplicación:
   a. Si la aplicación emite la llamada MQDISC antes de finalizar:
      - Para una unidad de trabajo coordinada del gestor de colas, el gestor de colas emite la llamada MQCMIT en nombre de la aplicación. La unidad de trabajo se confirma si es posible y se restituye si no es así.
      - Para una unidad de trabajo coordinada externamente, no hay ningún cambio en el estado de la unidad de trabajo; sin embargo, el gestor de colas indicará que la unidad de trabajo debe confirmarse, cuando lo solicite el coordinador de la unidad de trabajo.
   b. Si la aplicación finaliza normalmente pero sin emitir la llamada MQDISC, la unidad de trabajo se restituye.
   c. Si la aplicación finaliza *anormalmente* sin emitir la llamada MQDISC, la unidad de trabajo se restituye.

Parámetros

La llamada MQDISC tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada/salida

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

Al finalizar correctamente la llamada, el gestor de colas establece *HCONN* en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

HCUNUH
Descriptor de conexión inutilizable.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDISC(HCONN : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Suprimir descriptor de mensaje) en IBM i

La llamada MQDLTMH suprime un manejador de mensajes y es el inverso de la llamada MQCRTMH.

- "Sintaxis" en la página 1323
- "Notas de uso" en la página 1323
- "Parámetros" en la página 1324
- "Declaración RPG" en la página 1326

Sintaxis

MQDLTMH *((Hconn, Hmsg, DltMsgHOpts, CompCode, Reason)*

Notas de uso

1. Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
   - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
   - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de IBM MQ .

   Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte "MQBEGIN (Iniciar unidad de trabajo) en IBM i" en la página 1283.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
   - En z/OS, utilice las llamadas siguientes:
     - Los programas por lotes (incluidos los programas IMS batch DL/I) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de IBM MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de IBM MQ como a los recursos pertenecientes a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
     - Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT ROLLBACK` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS .
     - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como `ROLB` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).
   - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **`CMTSCOPE(*JOB)`** no debe haberse emitido para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC (Desconectar gestor de colas) en IBM i" en la página 1321 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *`GroupId`*, *`MsgSeqNumber`*, *`Offset`*y *`MsgFlags`* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

El gestor de colas mantiene tres conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:

- La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).

Si la aplicación coloca u obtiene los mensajes como parte de una unidad de trabajo y, a continuación, la aplicación restituye la unidad de trabajo, la información de grupo y segmento se restaura al valor que tenía anteriormente:

- La información asociada con la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT satisfactoria para ese manejador de cola en la unidad de trabajo actual.
- La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo. El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en PMOPT (entero con signo de 10 dígitos)y la opción GMLOGO descrita en GMOPT (entero con signo de 10 dígitos).

Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:

5. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte HCONN (entero con signo de 10 dígitos)-salida para obtener información sobre el ámbito de los manejadores de conexión.
6. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
7. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse de esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQDLTMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG** .

Si el descriptor de contexto de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de contexto de mensaje; de lo contrario, la llamada falla con RC2009 .

HMSG (entero con signo de 20 dígitos)-entrada/salida

Este es el descriptor de contexto de mensaje que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

Al finalizar correctamente la llamada, el descriptor de contexto se establece en un valor no válido para el entorno. Este valor es:

HMUNUH
Manejador de mensajes inutilizable.

El descriptor de mensaje no se puede suprimir si hay otra llamada IBM MQ en curso a la que se ha pasado el mismo descriptor de mensaje.

DLTOPT (MQDMHO)-entrada

Consulte MQDMHO para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2009
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2462
(2462, X'099E') La estructura de opciones del manejador de mensajes no es válida.

RC2460
(2460, X'099C') El puntero de manejador de mensajes no es válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Consulte “Códigos de retorno para IBM i (ILE RPG)” en la página 1462 para obtener más detalles.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMH(HCONN : HMSG : DLTOPT :
                                   CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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-Suprimir propiedad de mensaje

La llamada MQDLTMP suprime una propiedad de un manejador de mensajes y es la inversa de la llamada MQSETMP.

- “Sintaxis” en la página 1326
- “Parámetros” en la página 1326
- “Declaración RPG” en la página 1328

Sintaxis

MQDLTMP *(Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason)*

Parámetros

La llamada MQDLTMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-Entrada

Este manejador representa la conexión con el gestor de colas. El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG** .

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje, de lo contrario la llamada fallará con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este es el descriptor de contexto de mensaje que contiene la propiedad que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

DLTOPT (MQDMPO)-Entrada

Consulte el tipo de datos MQDMPO para obtener detalles.

PRNAME (MQCHARV)-entrada

El nombre de la propiedad que se va a suprimir. Consulte Nombres de propiedad para obtener más información sobre los nombres de propiedad.

Los comodines no están permitidos en el nombre de propiedad.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2471
(2471, X'09A7') Propiedad no disponible.

RC2421
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

RC2157
(2157, X'086D') Los ASID primario y de inicio difieren.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2009
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2481
(2481, X'09B1') Suprimir estructura de opciones de propiedad de mensaje no válida.

RC2460
(2460, X'099C') Descriptor de mensaje no válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2442
(2442, X'098A') Nombre de propiedad no válido.

RC2111
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

RC2195
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener más información sobre estos códigos, consulte Códigos de terminación y razón de la API.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMP(HCONN : HMSG : DLTOPT :
                                   PRNAME : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Obtener mensaje) en IBM i

La llamada MQGET recupera un mensaje de una cola local que se ha abierto utilizando la llamada MQOPEN.

- “Sintaxis” en la página 1328
- “Notas de uso” en la página 1328
- “Parámetros” en la página 1331
- “Declaración RPG” en la página 1336

Sintaxis

MQGET *(HCONN, HOBJ, MSGDSC, GMO, BUFLEN, BUFFER, DATLEN, CMPCOD, REASON)*

Notas de uso

1. El mensaje recuperado se suprime normalmente de la cola. Esta supresión puede producirse como parte de la propia llamada MQGET o como parte de un punto de sincronización. La supresión de mensajes no se produce si se especifica una opción GMBRWF o GMBRWN en el parámetro **GMO** (consulte el campo *GMOPT* descrito en “MQGMO (Opciones de obtención de mensajes) en IBM i” en la página 1098).
2. Si se especifica la opción GMLK con una de las opciones de examen, el mensaje examinado se bloquea para que sea visible sólo para este descriptor de contexto.

   Si se especifica la opción GMUNLK, se desbloquea un mensaje bloqueado anteriormente. No se recupera ningún mensaje en este caso y los parámetros **MSGDSC**, **BUFLEN**, **BUFFER** y **DATLEN** no se comprueban ni modifican.
3. Si la aplicación que emite la llamada MQGET se ejecuta como un IBM MQ MQI client, es posible que el mensaje recuperado se pierda si durante el proceso de la llamada MQGET el IBM MQ MQI client termina de forma anómala o se interrumpe la conexión de cliente. Esto se debe a que el sustituto que se ejecuta en la plataforma del gestor de colas y que emite la llamada MQGET en nombre del cliente

no puede detectar la pérdida del cliente hasta que el sustituto está a punto de devolver el mensaje al cliente; esto es después de que el mensaje se haya eliminado de la cola. Esto puede ocurrir tanto para mensajes persistentes como para mensajes no persistentes.

El riesgo de perder mensajes de esta forma se puede eliminar recuperando siempre mensajes dentro de las unidades de trabajo (es decir, especificando la opción GMSYP en la llamada MQGET y utilizando las llamadas MQCMIT o MQBACK para confirmar o restituir la unidad de trabajo cuando se complete el proceso del mensaje). Si se especifica GMSYP y el cliente termina de forma anómala o se interrumpe la conexión, el sustituto restituye la unidad de trabajo en el gestor de colas y el mensaje se restablece en la cola.

En principio, se puede producir la misma situación con las aplicaciones que se están ejecutando en la plataforma del gestor de colas, pero en este caso la ventana durante la que se puede perder un mensaje es pequeña. Sin embargo, al igual que con IBM MQ MQI clients , el riesgo se puede eliminar recuperando el mensaje dentro de una unidad de trabajo.

4. Si una aplicación pone una secuencia de mensajes en unadeterminada cola dentro de una única unidad de trabajo y, a continuación, confirma esa unidad de trabajo de forma satisfactoria, los mensajes están disponibles para la recuperación tal como se indica a continuación:
   - Si la cola es una *cola no compartida* (es decir, una cola local), todos los mensajes de la unidad de trabajo pasan a estar disponibles al mismo tiempo.
   - Si la cola es una *cola compartida*, los mensajes de la unidad de trabajo pasan a estar disponibles en el orden en el que se colocaron, pero no todos al mismo tiempo. Cuando el sistema está muy cargado, es posible que el primer mensaje de la unidad de trabajo se recupere correctamente, pero que la llamada MQGET para el segundo o posterior mensaje de la unidad de trabajo falle con RC2033. Si esto ocurre, la aplicación debe esperar un poco y luego volver a intentar la operación.
5. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Consulte las notas de uso en la descripción de la llamada MQPUT para obtener detalles. Si las condiciones se cumplen, los mensajes se presentan a la aplicación receptora en el orden en que fueron enviados, si:
   - Sólo un receptor está obteniendo mensajes de la cola.

     Si hay dos o más aplicaciones que obtienen mensajes de la cola, debe ponerse de acuerdo con el remitente sobre el mecanismo que se utilizará para identificar los mensajes que pertenecen a una secuencia. Por ejemplo, el remitente puede establecer todos los campos MDCID de los mensajes de una secuencia en un valor que sea exclusivo para esa secuencia de mensajes.
   - El receptor no cambia deliberadamente el orden de recuperación, por ejemplo, especificando un MDMID o MDCIDdeterminado.

   Si la aplicación emisora coloca los mensajes como un grupo de mensajes, los mensajes se presentan a la aplicación receptora en el orden correcto si la aplicación receptora especifica la opción GMLOGO en la llamada MQGET. Para obtener más información sobre los grupos de mensajes, consulte:
   - MDMFL Campo de MQMD
   - Opción PMLOGO en MQPMO
   - Opción GMLOGO en MQGMO
6. Las aplicaciones prueban el código de comentarios FBQUIT en el campo MDFB del parámetro **MSGDSC** . Si se encuentra este valor, la aplicación finaliza. Consulte el campo MDFB descrito en “MQMD (Descriptor de mensaje) en IBM i” en la página 1133 para obtener más información.
7. Si la cola identificada por HOBJ se ha abierto con la opción OOSAVA, y el código de finalización de la llamada MQGET es CCOK o CCWARN, el contexto asociado con el descriptor de contexto de cola HOBJ se establece en el contexto del mensaje que se ha recuperado (a menos que se establezca la opción GMBRWF o GMBRWN, en cuyo caso el contexto se marca como no disponible). Este contexto se puede utilizar en una llamada MQPUT o MQPUT1 posterior especificando las opciones PMPASI o PMPASA. Esto permite que el contexto del mensaje recibido se transfiera en su totalidad o en parte a otro mensaje (por ejemplo, cuando el mensaje se reenvía a otra cola). Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

8. Si la opción GMCONV se incluye en el parámetro **GMO** , los datos del mensaje de aplicación se convierten a la representación solicitada por la aplicación receptora, antes de que los datos se coloquen en el parámetro **BUFFER** :
   - El campo MDFMT de la información de control del mensaje identifica la estructura de los datos de aplicación y los campos MDCSI y MDENC de la información de control del mensaje especifican su identificador y codificación de juego de caracteres.
   - La aplicación que emite la llamada MQGET especifica en los campos MDCSI y MDENC del parámetro **MSGDSC** el identificador de juego de caracteres y la codificación a los que se deben convertir los datos del mensaje de aplicación.

   Cuando es necesaria la conversión de los datos del mensaje, la conversión la realiza el propio gestor de colas o una salida escrita por el usuario, en función del valor del campo MDFMT en la información de control del mensaje:

   - El gestor de colas convierte automáticamente los formatos siguientes; estos formatos se denominan formatos "incorporados":

FMADMN	FMMDE
FMCICS	FMPCF
FMCMD1	HMRM
FMCMD2	FMRFH
FMDLH	FMRFH2
FMDH	FMSTR
FMEVNT	FMTM
FMIMS	FMXQH
FMIMVS	

   - El nombre de formato FMNONE es un valor especial que indica que la naturaleza de los datos del mensaje no está definida. Como consecuencia, el gestor de colas no intenta la conversión cuando se recupera el mensaje de la cola.

     Nota: Si se especifica GMCONV en la llamada MQGET para un mensaje que tiene un nombre de formato FMNONE, y el juego de caracteres o la codificación del mensaje difiere del especificado en el parámetro **MSGDSC** , el mensaje se devuelve en el parámetro **BUFFER** (suponiendo que no haya otros errores), pero la llamada se completa con el código de terminación CCWARN y el código de razón RC2110.

     FMNONE puede utilizarse cuando la naturaleza de los datos del mensaje significa que no requiere conversión, o cuando las aplicaciones de envío y recepción han acordado entre sí el formulario en el que deben enviarse los datos del mensaje.
   - Todos los demás nombres de formato hacen que el mensaje se pase a una salida escrita por el usuario para su conversión. La salida tiene el mismo nombre que el formato, aparte de las adiciones específicas del entorno. Los nombres de formato especificados por el usuario no deben empezar por las letras "MQ", ya que estos nombres pueden entrar en conflicto con los nombres de formato soportados en el futuro.

   Los datos de usuario del mensaje se pueden convertir entre cualquier juego de caracteres y codificación admitidos. Sin embargo, tenga en cuenta que si el mensaje contiene una o más estructuras de cabecera IBM MQ , el mensaje no se puede convertir de o a un juego de caracteres que tenga caracteres de doble byte o de varios bytes para cualquiera de los caracteres válidos en los nombres de cola. El código de razón RC2111 o RC2115 da como resultado si se intenta y el mensaje se devuelve sin convertir. El juego de caracteres Unicode UTF-16 es un ejemplo de este tipo de juego de caracteres.

   En la devolución de la llamada MQGET, el código de razón siguiente indica que el mensaje se ha convertido correctamente:

- RCNONE

El siguiente código de razón indica que el mensaje puede haberse convertido correctamente; la aplicación debe comprobar los campos `MDCSI` y `MDENC` en el parámetro **MSGDSC** para averiguarlo:

- RC2079

Los demás códigos de razón indican que el mensaje no se ha convertido.

Nota: La interpretación del código de razón descrito en este ejemplo es verdadera para las conversiones realizadas por salidas escritas por el usuario sólo si la salida se ajusta a las directrices de proceso.

9. Para los formatos incorporados listados anteriormente, el gestor de colas puede realizar la conversión predeterminada de series de caracteres en el mensaje cuando se especifica la opción GMCONV. La conversión predeterminada permite que, para convertir datos de series de caracteres, el gestor de colas utilice un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de conjunto de caracteres real. Como resultado, la llamada MQGET puede ejecutarse correctamente con el código de terminación CCOK, en lugar de completarse con CCWARN y el código de razón RC2111 o RC2115.

   Nota: El resultado de utilizar un juego de caracteres aproximado para convertir datos de series de caracteres es que es posible que algunos se conviertan de forma incorrecta. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres real como al juego de caracteres predeterminado.

   La conversión predeterminada se aplica a los datos de los mensajes de la aplicación y a los campos de tipo carácter de las estructuras MQMD y MQMDE:

   - La conversión predeterminada de los datos de mensaje de aplicación sólo se produce cuando se cumplen todas las sentencias siguientes:
     - La aplicación especifica GMCONV.
     - El mensaje contiene datos que se deben convertir de o a un juego de caracteres que no está soportado.
     - La conversión predeterminada no estaba activada cuando se ha instalado o reiniciado el gestor de colas.
   - La conversión predeterminada de los campos de caracteres de las estructuras MQMD y MQMDE se produce según convenga, si dicha conversión predeterminada está habilitada para el gestor de colas. La conversión se realiza incluso si la aplicación no especifica la opción GMCONV en la llamada MQGET.

10. El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico.

    Declarar el parámetro como una estructura aumenta la longitud máxima posible a 9999 bytes, mientras que declarar el parámetro como un campo en un archivo físico aumenta la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQGET tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de `HCONN` ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este manejador representa la cola de la que debe recuperarse el mensaje. El valor de HOBJ ha sido devuelto por una llamada MQOPEN anterior. La cola debe haberse abierto con una o más de las opciones siguientes (consulte "MQOPEN (Abrir objeto) en IBM i" en la página 1353 para obtener más detalles):

- OOINPS
- OOINPX
- OOINPQ
- OOBRW

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado. Consulte "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener los detalles.

Si BUFLEN es menor que la longitud del mensaje, el gestor de colas sigue especificando MSGDSC , si se ha especificado GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098).

Si la aplicación proporciona un MQMD version-1 , el mensaje devuelto tiene un MQMDE con el prefijo de los datos del mensaje de aplicación, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. Si todos los campos de MQMDE tienen valores predeterminados, se omite MQMDE. Un nombre de formato FMMDE en el campo MDFMT en MQMD indica que hay un MQMDE presente.

GMO (MQGMO)-entrada/salida

Opciones que controlan la acción de MQGET.

Consulte "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener los detalles.

BUFLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes del área BUFFER .

Se puede especificar cero para los mensajes que no tienen datos, o si el mensaje se va a eliminar de la cola y los datos descartados (en este caso, se debe especificar GMATM).

Nota: La longitud del mensaje más largo que es posible leer de la cola la proporciona el atributo de cola **MaxMsgLength** ; consulte "Atributos para colas" en la página 1402.

BUFFER (serie de bits de 1 byte x BUFLEN)-salida

Área para contener los datos del mensaje.

El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si BUFLEN es menor que la longitud del mensaje, la mayor parte posible del mensaje se mueve a BUFFER ; esto sucede si se especifica GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener más información).

El juego de caracteres y la codificación de los datos en **BUFFER** se proporcionan mediante los campos MDCSI y MDENC devueltos en el parámetro **MSGDSC** . Si estos valores son distintos de los valores que necesita el destinatario, éste deberá convertir los datos de mensajes de la aplicación en el juego de caracteres y en la codificación necesarios. La opción GMCONV se puede utilizar con una salida escrita por el usuario para realizar la conversión de los datos del mensaje (consulte "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener detalles de esta opción).

Nota: Todos los demás parámetros de la llamada MQGET están en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas **CodedCharSetId** y ENNAT).

Si falla la llamada, el contenido del almacenamiento también se podría haber modificado.

DATLEN (entero con signo de 10 dígitos)-salida

Longitud del mensaje.

Es la longitud en bytes de los datos de aplicación del mensaje. Si esta longitud de mensaje es mayor que BUFLEN, sólo se devuelven BUFLEN bytes en el parámetro **BUFFER** (es decir, el mensaje se trunca). Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

Si BUFLEN es menor que la longitud del mensaje, el gestor de colas sigue especificando DATLEN , si se ha especificado GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener más información). Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para introducir los datos del mensaje y después reemitir la llamada con un almacenamiento intermedio que tenga el tamaño adecuado.

Sin embargo, si se especifica la opción GMCONV y los datos de mensaje convertidos son demasiado largos para caber en BUFFER, el valor devuelto para DATLEN es:

- La longitud de los datos no convertidos, para los formatos definidos por el gestor de colas.

  En este caso, si la naturaleza de los datos hace que se expanda durante la conversión, la aplicación debe asignar un almacenamiento intermedio mayor que el valor devuelto por el gestor de colas para DATLEN.
- El valor devuelto por la salida de conversión de datos, para formatos definidos por la aplicación.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
: Realización satisfactoria.

CCWARN
: Aviso (finalización parcial).

CCFAIL
: La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Los siguientes códigos de razón son los que el gestor de colas puede devolver para el parámetro **REASON** . Si la aplicación especifica la opción GMCONV y se invoca una salida escrita por el usuario para convertir algunos o todos los datos del mensaje, es la salida la que decide qué valor se devuelve para el parámetro **REASON** . Como resultado, los valores que no sean los documentados más adelante en esta sección son posibles.

Si CMPCOD es CCOK:

RCNONE
: (0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCWARN:

RC2120
: (2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

RC2190
: (2190, X'88E') La serie convertida es demasiado grande para el campo.

RC2150
(2150, X'866') La serie DBCS no es válida.

RC2110
(2110, X'83E') El formato de mensaje no válido.

RC2243
(2243, X'8C3') Los segmentos del mensaje tienen distintos CCSID.

RC2244
(2244, X'8C4') Los segmentos del mensaje tienen distintas codificaciones.

RC2209
(2209, X'8A1') No hay ningún mensaje bloqueado.

RC2119
(2119, X'847') Los datos del mensaje no se han convertido.

RC2272
(2272, X'8E0') Los datos del mensaje se han convertido parcialmente.

RC2145
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

RC2111
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2113
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

RC2114
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

RC2112
(2112, X'840') Codificación de entero de origen no reconocida.

RC2143
(2143, X'85F') Parámetro de longitud de origen no válido.

RC2146
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

RC2115
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2117
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

RC2118
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

RC2116
(2116, X'844') Codificación de entero de destino no reconocida.

RC2079
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

RC2080
(2080, X'820') Se ha devuelto un mensaje truncado (el proceso no se ha completado).

Si CMPCOD es CCFAIL:

RC2004
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2010
(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2016
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2186
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2241
(2241, X'8C1') El grupo de mensajes no está completo.

RC2242
(2242, X'8C2') El mensaje lógico no está completo.

RC2259
(2259, X'8D3') La especificación de examinar es incoherente.

RC2245
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2247
(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2026
(2026, X'7EA') El descriptor de mensaje no es válido.

RC2250
(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2033
(2033, X'7F1') No hay ningún mensaje disponible.

RC2034
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

RC2036
(2036, X'7F4') La cola no se ha abierto para examen.

RC2037
(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2024
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2072
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2255
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                                     BUFLEN : BUFFER : DATLEN :
C                                     CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Consultar sobre atributos de objeto) en IBM i

La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

Son válidos los siguientes tipos de objeto:

- Cola
- Lista de nombres

- Definición de proceso
- Gestor de colas
- “Sintaxis” en la página 1337
- “Notas de uso” en la página 1337
- “Parámetros” en la página 1338
- “Declaración RPG” en la página 1345

Sintaxis

MQINQ *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Notas de uso

1. Los valores devueltos son una instantánea de los atributos seleccionados. No hay garantía de que los atributos no se modifiquen antes de que la aplicación pueda actuar sobre los valores devueltos.
2. Al abrir una cola modelo, se crea una cola local dinámica. Esto es cierto incluso si abre la cola modelo para consultar sus atributos.

   Los atributos de la cola dinámica (con determinadas excepciones) son los mismos que los de la cola modelo en el momento en que se crea la cola dinámica. Si luego utiliza la llamada MQINQ en esta cola, el gestor de colas devuelve los atributos de la cola dinámica y no los de la cola modelo. Consulte la Tabla 1 para obtener detalles sobre qué atributos de la cola modelo hereda la cola dinámica.
3. Si el objeto que se está consultando es una cola alias, los valores de atributo devueltos por la llamada MQINQ son los de la cola alias y no los de la cola base en la que se resuelve el alias.
4. Si el objeto que se está consultando es una cola de clúster, los atributos que se pueden consultar dependen de cómo se abra la cola:
   - Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria. En este caso, los atributos que se pueden consultar son los válidos para las colas locales.
   - Si la cola de clúster se abre sólo para consultas, o para consultas y salidas, sólo se pueden consultar los atributos siguientes; el atributo **QType** tiene el valor QTCLUS en este caso:
     - CAQD
     - CQN
     - IADBND
     - IADPER
     - IADPRI
     - IIPUT
     - IQTYP

     Si la cola de clúster se abre sin ningún enlace fijo (es decir, OOBNDN especificado en la llamada MQOPEN, o OOBNDQ especificado cuando el atributo **DefBind** tiene el valor BNDNOT), las llamadas MQINQ sucesivas para la cola pueden consultar distintas instancias de la cola de clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

   Para obtener más información sobre las colas de clúster, consulte Configuración de un clúster de gestores de colas.
5. Si se van a consultar varios atributos y, a continuación, algunos de ellos se van a establecer utilizando la llamada MQSET, puede ser conveniente situar al principio de las matrices de selector los atributos que se van a establecer, de forma que se puedan utilizar las mismas matrices (con recuentos reducidos) para MQSET.
6. Si surge más de una de las situaciones de aviso (consulte el parámetro **CMPCOD**), el código de razón devuelto es el *primero* de la lista siguiente que se aplica:

a. RC2068
b. RC2022
c. RC2008

7. Para obtener más información sobre los atributos de objeto, consulte:
   - “Atributos para colas” en la página 1402
   - “Atributos de las listas de nombres” en la página 1431
   - “Atributos para definiciones de proceso en IBM i” en la página 1432
   - “Atributos del gestor de colas en IBM i” en la página 1434
8. Una cola local nueva SYSTEM.ADMIN.COMMAND.EVENT se utiliza para poner en cola los mensajes que se generan siempre que se emiten mandatos. Los mensajes se colocan en esta cola para la mayoría de los mandatos, en función de cómo se establezca el atributo de gestor de colas CMDEV:
   - ENABLED-los mensajes de suceso de mandato se generan y se colocan en la cola para todos los mandatos satisfactorios.
   - NODISPLAY-los mensajes de suceso de mandato se generan y se colocan en la cola para todos los mandatos satisfactorios que no sean el mandato DISPLAY (MQSC) y el mandato Inquire (PCF).
   - DISABLED-los mensajes de suceso de mandato no se generan (este es el valor predeterminado inicial del gestor de colas).

Parámetros

La llamada MQINQ tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este descriptor de contexto representa el objeto (de cualquier tipo) con los atributos necesarios. El descriptor de contexto debe haber sido devuelto por una llamada MQOPEN anterior que haya especificado la opción OOINQ.

SELCNT (entero con signo de 10 dígitos)-entrada

Recuento de selectores.

Este es el recuento de selectores que se proporcionan en la matriz *SELS* . Es el número de atributos que se van a devolver. Cero es un valor válido. El número máximo permitido es 256.

SELS (entero con signo de 10 dígitos x SELCNT)-entrada

Matriz de selectores de atributos.

Esta es una matriz de selectores de atributos **SELCNT** ; cada selector identifica un atributo (entero o carácter) con un valor que es necesario.

Cada selector debe ser válido para el tipo de objeto que representa *HOBJ* ; de lo contrario, la llamada falla con el código de terminación CCFAIL y el código de razón RC2067.

En el caso especial de las colas:

- Si el selector no es válido para colas de *cualquier* tipo, la llamada falla con el código de terminación CCFAIL y el código de razón RC2067.
- Si el selector sólo es aplicable a colas de tipo o tipos distintos al del objeto, la llamada se realiza correctamente con el código de terminación CCWARN y el código de razón RC2068.

- Si la cola que se está consultando es una cola de clúster, los selectores que son válidos dependen de cómo se haya resuelto la cola; consulte la nota de uso 4 para obtener más detalles.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectores IA*) se devuelven en *INTATR* en el mismo orden en el que aparecen estos selectores en *SELS*. Los valores de atributo que corresponden a selectores de atributo de carácter (selectores CA*) se devuelven en *CHRATR* en el mismo orden en el que se producen dichos selectores. Los selectores IA* se pueden intercalar con los selectores CA*; sólo es importante el orden relativo dentro de cada tipo.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores IA* residen dentro del rango IAFRST a IALAST, y los selectores CA* dentro del rango CAFRST a CALAST.

   Para cada rango, las constantes IALSTU y CALSTU definen el valor más alto que acepta el gestor de colas.
2. Si todos los selectores IA* aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices *SELS* y *INTATR* .

Los atributos que se pueden consultar se listan en las tablas siguientes. Para los selectores CA*, la constante que define la longitud en bytes de la serie resultante en *CHRATR* se especifica entre paréntesis.

Tabla 746. Selectores de atributos MQINQ para colas

Selector	Descripción	Nota
CAALTD	Fecha de la modificación más reciente (LNDATE).	1
CAALTT	Hora de la modificación más reciente (LNTIME).	1
CBRQN	Nombre de reposición en cola de retroceso excesivo (LNQN).	5
CBASQ	Nombre de la cola en la que se resuelve el alias (LNQN).	
CACFSN	Nombre de estructura de recurso de acoplamiento (LNCFSN).	3
CACLN	Nombre de clúster (LNCLUN).	1
CCLNL	Lista de nombres de clúster (LNNLN).	1
CACRTD	Fecha de creación de cola (LNCRTD).	
CACRTT	Tiempo de creación de cola (LNCRTT).	
CINIQ	Nombre de cola de inicio (LNQN).	
CAPRON	Nombre de la definición de proceso (LNPRON).	
CAQD	Descripción de cola (LNQD).	
CQN	Nombre de cola (LNQN).	
CARQMN	Nombre del gestor de colas remoto (LNQMN).	
CARQN	Nombre de la cola remota tal como se conoce en el gestor de colas remoto (LNQN).	
CATRGD	Datos de desencadenante (LNTRGD).	5
CXQN	Nombre de cola de transmisión (LNQN).	
IABTHR	El umbral de restituciones.	5
CIDEP	Número de mensajes en cola.	

Tabla 746. Selectores de atributos MQINQ para colas (continuación)		
Selector	**Descripción**	**Nota**
IADBND	Enlace predeterminado.	1
IADINP	Opción de apertura para entrada predeterminada.	5
IADPER	Persistencia de mensajes predeterminada.	
IADPRI	Prioridad de mensaje predeterminada.	5
IADEFT	El tipo de definición de la cola.	
IADIST	Soporte de lista de distribución.	2
IHGB	Indica si se debe reforzar el recuento de restituciones.	5
IAIGET	Si se permiten operaciones get.	
IIPUT	Si se permiten las operaciones de colocación.	
IAMLEN	La longitud máxima de mensajes.	
IAMDEP	Número máximo de mensajes permitidos en la cola.	
IAMDS	Indica si la prioridad del mensaje es relevante.	5
OICI	Número de llamadas MQOPEN que tienen la cola abierta para entrada.	
IOOC	Número de llamadas MQOPEN que tienen la cola abierta para salida.	
IQDHE	Atributo de control para sucesos de profundidad de cola alta.	4, 5
IAQDHL	Límite alto para la profundidad de cola.	4, 5
IQDLE	Atributo de control para sucesos de profundidad de cola baja.	4, 5
IQDLL	Límite bajo para profundidad de cola.	4, 5
IQDME	Atributo de control para sucesos máximos de profundidad de cola.	4, 5
IQSI	Límite para el intervalo de servicio de cola.	4, 5
IQSIE	Atributo de control para sucesos de intervalo de servicio de cola.	4, 5
IQTYP	El tipo de cola.	
IAQSGD	Disposición del grupo de compartición de colas.	3
IARINT	Intervalo de retención de cola.	5
IASCOP	Ámbito de definición de cola.	4, 5
IASHAR	Si la cola se puede compartir para entrada.	
IATRGC	Control de desencadenante.	
IATRGD	Profundidad del desencadenante.	5
IATRGP	Umbral de prioridad del mensaje para activaciones.	5
IATRGT	Tipo de desencadenante.	
SAIUSAG	Uso.	
CLWLUSEQ	Utilice colas remotas.	

Nota:

1. Soportado en las plataformas siguientes:

- AIX AIX
- IBM i IBM i
- Solaris Solaris
- Windows Windows
- z/OS z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

2. Soportado en las plataformas siguientes:
   - AIX AIX
   - IBM i IBM i
   - Solaris Solaris
   - Windows Windows

   y para clientes de IBM MQ conectados a estos sistemas.
3. z/OS Soportado en z/OS.
4. z/OS No soportado en z/OS.
5. No soportado en VSE/ESA.

Tabla 747. Selectores de atributos MQINQ para listas de nombres

Selector	Descripción	Nota
CAALTD	Fecha de la modificación más reciente (LNDATE)	1
CAALTT	Hora de la modificación más reciente (LNTIME)	1
CALSTD	Descripción de lista de nombres (LNNLD)	1
CALSTN	Nombre del objeto de lista de nombres (LNNLN)	1
CANAMS	Nombres de la lista de nombres (LNQN x *Número de nombres de la lista)*	1
IANAMC	Número de nombres en la lista de nombres	1
IAQSGD	Disposición de grupo de uso compartido de colas	3

Tabla 748. Selectores de atributos MQINQ para definiciones de proceso

Selector	Descripción	Nota
CAALTD	Fecha de la modificación más reciente (LNDATE)	1
CAALTT	Hora de la modificación más reciente (LNTIME)	1
CAAPPI	Identificador de aplicación (LNPROA)	5
CAENVD	Datos de entorno (LNPROE)	5
CAPROD	Descripción de la definición de proceso (LNPROD)	5
CAPRON	Nombre de definición de proceso (LNPRON)	5
CAUSRD	Datos de usuario (LNPROU)	5
IAPPT	Tipo de aplicación	5
IAQSGD	Disposición de grupo de uso compartido de colas	3

Tabla 749. Selectores de atributos MQINQ para el gestor de colas		
Selector	**Descripción**	**Nota**
CAALTD	Fecha de la modificación más reciente (LNDATE)	1
CAALTT	Hora de la modificación más reciente (LNTIME)	1
CACADX	Nombre de salida de definición de canal automática (LNEXN)	1
CACLWD	Datos pasados a la salida de carga de trabajo de clúster (LNEXDA)	1
CACLWX	Nombre de salida de carga de trabajo de clúster (LNEXN)	1
CACMDQ	Nombre de cola de entrada de mandatos del sistema (LNQN)	5
CADLQ	Nombre de la cola de mensajes no entregados (LNQN)	5
CADXQN	Nombre de cola de transmisión predeterminada (LNQN)	5
CAQMD	Descripción del gestor de colas (LNQMD)	5
CAQMID	Identificador de gestor de colas (LNQMID)	1
CQMN	Nombre del gestor de colas local (LNQMN)	5
CAQSGN	Nombre de grupo de compartición de colas (LNQSGN)	3
CARPN	Nombre del clúster para el que el gestor de colas proporciona servicios de repositorio (LNQMN)	1
CARPNL	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que el gestor de colas proporciona servicios de repositorio (LNNLN)	1
CMDEV	Atributo de control que determina si los mensajes generados cuando se emiten mandatos se colocan en una cola	8
IAAUTE	Atributo de control para sucesos de autorización	4, 5
ICAD	Atributo de control para definición de canal automática	2
IIACADE	Atributo de control para sucesos de definición de canal automática	2
CILXQ	Tipo de cola de transmisión de clúster predeterminado	4
LLWL	Longitud de la carga de trabajo de clúster	1
SIGC	Identificador de juego de caracteres codificado	5
CIADMDL	Nivel de mandatos soportado por el gestor de colas	5
AICFGE	Atributo de control para sucesos de configuración	3
IADIST	Soporte de lista de distribución	2
IINHE	Atributo de control para sucesos de inhibición	4, 5
IALCLE	Atributo de control para sucesos locales	4, 5
IAMHND	Número máximo de descriptores de contexto	5
IAMLEN	Longitud máxima de mensaje	5
IAMPRI	Prioridad máxima	5
IAMUNC	Número máximo de mensajes no confirmados dentro de una unidad de trabajo	5
IAPFME	Atributo de control para sucesos de rendimiento	4, 5
IAPLAT	Plataforma en la que reside el gestor de colas	5

Tabla 749. Selectores de atributos MQINQ para el gestor de colas (continuación)		
Selector	**Descripción**	**Nota**
IARMTE	Atributo de control para sucesos remotos	4, 5
IASSO	Atributo de control para sucesos de detención de inicio	4, 5
IASYNC	Disponibilidad de punto de sincronización	5
IATRLFT	Duración de los temas no administrativos no utilizados	
IATRGI	Activar intervalo	5

IACNT (entero con signo de 10 dígitos)-entrada

Recuento de atributos enteros.

Es el número de elementos de la matriz INTATR . Cero es un valor válido.

Si es como mínimo el número de selectores IA* en el parámetro **SELS** , se devuelven todos los atributos enteros solicitados.

INTATR (entero con signo de 10 dígitos x IACNT)-salida

Matriz de atributos enteros.

Esta es una matriz de valores de atributo de entero de *IACNT* .

Los valores de atributos enteros se devuelven en el mismo orden que los selectores IA* en el parámetro **SELS** . Si la matriz contiene más elementos que el número de selectores IA*, el exceso de elementos no se modificará.

Si HOBJ representa una cola, pero un selector de atributos no es aplicable a ese tipo de cola, se devuelve el valor específico IAVNA para el elemento correspondiente en la matriz INTATR .

CALEN (entero con signo de 10 dígitos)-entrada

Longitud del almacenamiento intermedio de atributos de caracteres.

Es la longitud en bytes del parámetro **CHRATR** .

Debe ser al menos la suma de las longitudes de los atributos de caracteres solicitados (consulte SELS). Cero es un valor válido.

CHRATR (serie de caracteres de 1 byte x CALEN)-salida

Atributos de carácter.

Es el almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenados entre sí. La longitud del almacenamiento intermedio la proporciona el parámetro **CALEN** .

Los atributos de carácter se devuelven en el mismo orden que los selectores CA* en el parámetro **SELS** . La longitud de cada serie de atributo es fija para cada atributo (consulte SELS), y el valor que contiene se rellena a la derecha con espacios en blanco si es necesario. Si el almacenamiento intermedio es mayor que el necesario para contener todos los atributos de caracteres solicitados (incluido el relleno), los bytes más allá del último valor de atributo devuelto no se modifican.

Si HOBJ representa una cola, pero un selector de atributos no es aplicable a ese tipo de cola, se devuelve una serie de caracteres que consta totalmente de asteriscos (*) como valor de ese atributo en CHRATR.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Si CMPCOD es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2008
(2008, X'7D8') No se permite espacio suficiente para los atributos de caracteres.

RC2022
(2022, X'7E6') No se permite espacio suficiente para atributos enteros.

RC2068
(2068, X'814 ') Selector no aplicable al tipo de cola.

Si *CMPCOD* es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2006
(2006, X'7D6') Longitud de atributos de caracteres no válida.

RC2007
(2007, X'7D7') Serie de atributos de carácter no válida.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2021
(2021, X'7E5') Recuento de atributos enteros no válidos.

RC2023
(2023, X'7E7') Matriz de atributos enteros no válida.

RC2038
(2038, X'7F6') Cola no abierta para consulta.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2065
(2065, X'811 ') Recuento de selectores no válido.

RC2067
(2067, X'813 ') El selector de atributos no es válido.

RC2066
(2066, X'812 ') Recuento de selectores demasiado grande.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQ(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

La definición de prototipo para la llamada es:

```
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 (propiedad Consultar mensaje) en IBM i

La llamada MQINQMP devuelve el valor de una propiedad de un mensaje.

- “Sintaxis” en la página 1345
- “Parámetros” en la página 1345
- “Declaración RPG” en la página 1350

Sintaxis

MQINQMP *(Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type, ValueLength, Value, DataLength, CompCode, Reason)*

Parámetros

La llamada MQINQMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg** .

Si el descriptor de contexto de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra consultando una propiedad del descriptor de contexto de mensaje; de lo contrario, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este es el manejador de mensajes que se debe consultar. El valor ha sido devuelto por una llamada **MQCRTMH** anterior.

INQOPT (MQIMPO)-entrada

Consulte el tipo de datos MQIMPO para obtener más detalles.

PRNAME (MQCHARV)-entrada

Describe el nombre de la propiedad a consultar.

Si no se encuentra ninguna propiedad con este nombre, la llamada falla con la razón RC2471.

Puede utilizar el carácter de signo de porcentaje (%) al final del nombre de propiedad. El comodín coincide con cero o más caracteres, incluido el carácter de punto (.). Esto permite a una aplicación consultar el valor de muchas propiedades. Llame a MQINQMP con la opción IPINQF para obtener la primera propiedad coincidente y de nuevo con la opción IPINQN para obtener la siguiente propiedad coincidente. Cuando no hay más propiedades coincidentes disponibles, la llamada falla con RC2471. Si el campo *ReturnedName* de la estructura InqPropOpts se inicializa con una dirección o un desplazamiento para el nombre devuelto de la propiedad, esto se completa al devolver MQINQMP con el nombre de la propiedad que ha coincidido. Si el campo *VSBufSize* de *ReturnedName* en la estructura de Opts InqPropes menor que la longitud del nombre de propiedad devuelto, el código de terminación se establece en CCFAIL con la razón RC2465.

Las propiedades que tienen sinónimos conocidos se devuelven de la siguiente manera:

1. Propiedades con el prefijo "mqps." se devuelven con el nombre de propiedad IBM MQ . Por ejemplo, "MQTopicString" es el nombre devuelto en lugar de "mqps.Top".
2. Propiedades con el prefijo "jms." o "mcd." se devuelven como nombre de campo de cabecera JMS . Por ejemplo, "JMSExpiration" es el nombre devuelto en lugar de "jms.Exp".
3. Propiedades con el prefijo "usr." se devuelven sin ese prefijo. Por ejemplo, se devuelve "Color" en lugar de "usr.Color".

Las propiedades con sinónimos sólo se devuelven una vez.

En el lenguaje de programación RPG, las variables de macro siguientes se definen para consultar todas las propiedades y todas las propiedades que empiezan por "usr.":

INQALL
Consultar todas las propiedades del mensaje.

INQUSR
Consultar todas las propiedades del mensaje que inician "usr.". El nombre devuelto se devuelve sin el "usr." .

Si se especifica IPINQN pero el nombre ha cambiado desde la llamada anterior o ésta es la primera llamada, entonces está implícito IPINQF.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PRPDSC (MQPD)-salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluido lo que sucede si la propiedad no está soportada, a qué contexto de mensaje pertenece la propiedad y en qué mensajes debe copiarse la propiedad. Consulte MQPD para obtener detalles de esta estructura.

TYPE (entero con signo de 10 dígitos)-entrada/salida

Al volver de la llamada MQINQMP, este parámetro se establece en el tipo de datos *Valor*. El tipo de datos puede ser cualquiera de los siguientes:

TYPBOL
Un booleano.

TYPBST
una serie de bytes.

TYPI8
Un entero con signo de 8 bits.

TYPI16
Un entero con signo de 16 bits.

TYPI32
Un entero con signo de 32 dígitos.

TYPI64
Un entero con signo de 64 bits.

TYPF32
Un número de coma flotante de 32 bits.

TYPF64
Un número de coma flotante de 64 bits.

TIPSTR
Una serie de caracteres.

TYPNUL
La propiedad existe pero tiene un valor nulo.

Si el tipo de datos del valor de propiedad no se reconoce, se devuelve TYPSTR y se coloca una representación de serie del valor en el área *Valor* . Se puede encontrar una representación de serie del tipo de datos en el campo *IPTYP* del parámetro *IPOPT* . Se devuelve un código de finalización de aviso con la razón RC2467.

Además, si se especifica la opción IPCTYP, se solicita la conversión del valor de propiedad. Utilice *Tipo* como entrada para especificar el tipo de datos con el que desea que se devuelva la propiedad. Consulte la descripción de la opción IPCTYP de “MQIMPO (Consultar opciones de propiedad de mensaje) en IBM i” en la página 1126 para obtener detalles de la conversión de tipo de datos.

Si no solicita la conversión de tipo, puede utilizar el valor siguiente en la entrada:

TYPASTO
El valor de la propiedad se devuelve sin convertir su tipo de datos.

VALLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes del área Valor.

Especifique cero para las propiedades para las que no necesita el valor devuelto. Estas podrían ser propiedades diseñadas por una aplicación para tener un valor nulo o una serie vacía. Especifique también cero si se ha especificado la opción IPQLEN; en este caso, no se devuelve ningún valor.

VALUE (bit de 1 byte stringxVALLEN)-salida

Este es el área que contiene el valor de propiedad consultado. El almacenamiento intermedio debe estar alineado en un límite adecuado para el valor que se devuelve. Si no lo hace, es posible que se produzca un error cuando se acceda más tarde al valor.

Si *VALLEN* es menor que la longitud del valor de propiedad, la mayor parte posible del valor de propiedad se mueve a *VALUE* y la llamada falla con el código de terminación CCFAIL y la razón RC2469.

El juego de caracteres de los datos en *VALUE* se proporciona mediante el campo IPRETCSI en el parámetro INQOPT. La codificación de los datos en *VALUE* la proporciona el campo IPRETENC en el parámetro INQOPT.

Si el parámetro *VALLEN* es cero, no se hace referencia a VALUE.

DATLEN (entero con signo de 10 dígitos)-salida

Es la longitud en bytes del valor de propiedad real tal como se devuelve en el área *Valor* .

Si *DataLength* es menor que la longitud del valor de la propiedad, *DataLength* se sigue entrando al devolver de la llamada MQINQMP. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad y, a continuación, volver a emitir la llamada con un almacenamiento intermedio del tamaño adecuado.

También se pueden devolver los valores siguientes.

Si el parámetro *Tipo* se establece en TYPSTR o TYPBST:

VLEMP
La propiedad existe pero no contiene caracteres ni bytes.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica a *CompCode*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es CCWARN:

RC2492
(2492, X'09BC') No se ha convertido el nombre de propiedad devuelto.

RC2466
(2466, X'09A2') Valor de propiedad no convertido.

RC2467
(2467, X'09A3') El tipo de datos de propiedad no está soportado.

RC2421
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

RC2157
(2157, X'086D') Los ASID primario y de inicio difieren.

RC2004
(2004, X'07D4') El parámetro de valor no es válido.

RC2005
(2005, X'07D5') El parámetro de longitud de valor no es válido.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2009
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2010
(2010, X'07DA') El parámetro de longitud de datos no es válido.

RC2464
(2464, X'09A0') Consultar estructura de opciones de propiedad de mensaje no válida.

RC2460
(2460, X'099C') Descriptor de mensaje no válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2064
(2046, X'07F8') Opciones no válidas o no coherentes.

RC2482
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

RC2470
(2470, X'09A6') La conversión del tipo de datos real al solicitado no está soportada.

RC2442
(2442, X'098A') Nombre de propiedad no válido.

RC2465
(2465, X'09A1') Nombre de propiedad demasiado grande para el almacenamiento intermedio de nombre devuelto.

RC2471
(2471, X'09A7) Propiedad no disponible.

RC2469
(2469, X'09A5') Valor de propiedad demasiado grande para el área Valor.

RC2472
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

RC2473
(2473, X'09A9') Tipo de propiedad solicitado no válido.

RC2111
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

RC2071
(2071, X'0871 ') Almacenamiento insuficiente disponible.

RC2195
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte:

- IBM MQ for z/OS mensajes, finalización, y códigos de razón para IBM MQ for z/OS
- Mensajes y códigos de razón para todas las demás plataformas IBM MQ

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQMP(HCONN : HMSG : INQOPT :
                                        PRNAME : PRPDSC : TYPE :
                                        VALLEN : VALUE : DATLEN :
                                        CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Convertir descriptor de mensaje en almacenamiento intermedio) en IBM i

MQMHBUF convierte un manejador de mensajes en un almacenamiento intermedio y es el inverso de la llamada MQBUFMH.

- “Sintaxis” en la página 1350
- “Notas de uso” en la página 1350
- “Parámetros” en la página 1351
- “Declaración RPG” en la página 1353

Sintaxis

MQMHBUF *(Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer, DataLength, CompCode, Reason)*

Notas de uso

MQMHBUF convierte un descriptor de mensaje en un almacenamiento intermedio.

Puede utilizarlo con una salida de API MQGET para acceder a determinadas propiedades, utilizando las API de propiedades de mensajes y, a continuación, volver a pasar estas propiedades en un almacenamiento intermedio a una aplicación diseñada para utilizar cabeceras MQRFH2 en lugar de manejadores de mensajes.

Esta llamada es la inversa de la llamada MQBUFMH, que puede utilizar para analizar propiedades de mensaje de un almacenamiento intermedio en un descriptor de mensaje.

Parámetros

La llamada MQMHBUF tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor de *HCONN* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG** .

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este descriptor de contexto es el descriptor de contexto de mensaje para el que se necesita un almacenamiento intermedio.

El valor ha sido devuelto por una llamada MQCRTMH anterior.

MHBOPT (MQMHBO)-entrada

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los almacenamientos intermedios a partir de los manejadores de mensajes.

Consulte "MQBMHO (opciones de almacenamiento intermedio a manejador de mensajes) en IBM i" en la página 1038 para obtener los detalles.

PRNAME (MQCHARV)-entrada

El nombre de la propiedad o propiedades que se van a poner en el almacenamiento intermedio.

Si no se encuentra ninguna propiedad que coincida con el nombre, la llamada falla con RC2471.

Comodines

Puede utilizar un comodín para colocar más de una propiedad en el almacenamiento intermedio. Para ello, utilice el signo de porcentaje (%) al final del nombre de propiedad. Este comodín coincide con cero o más caracteres, incluido el carácter de punto (.).

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

MSGDSC (MQMD)-entrada/salida

La estructura *MSGDSC* describe el contenido del área de almacenamiento intermedio.

En la salida, los campos *Encoding*, *CodedCharSetId* y *Format* se establecen para describir correctamente la codificación, el identificador de juego de caracteres y el formato de los datos en el área de almacenamiento intermedio tal como los graba la llamada.

Los datos de esta estructura están en el juego de caracteres y la codificación de la aplicación.

BUFLEN (entero con signo de 10 dígitos)-entrada

BUFFLEN es la longitud del área de almacenamiento intermedio, en bytes.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada/salida

BUFFER define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BUFFER* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro **MSGDSC** en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BUFLEN** es cero, no se hace referencia a *BUFFER* . En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DATLEN (entero con signo de 10 dígitos)-salida

DATLEN es la longitud, en bytes, de las propiedades devueltas en el almacenamiento intermedio. Si el valor es cero, ninguna propiedad coincide con el valor proporcionado en *PRNAME* y la llamada falla con el código de razón RC2471.

Si *BUFLEN* es menor que la longitud necesaria para almacenar las propiedades en el almacenamiento intermedio, la llamada MQMHBUF falla con RC2469, pero se sigue entrando un valor en *DATLEN*. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar las propiedades y, a continuación, volver a emitir la llamada con el *BUFLEN*necesario.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2501
(2501, X'095C') El descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio no es válido.

RC2004
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

RC2005
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2009
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2010
(2010, X'07DA') El parámetro de longitud de datos no es válido.

RC2460
(2460, X'099C') Descriptor de mensaje no válido.

RC2026
(2026, X'07EA') Descriptor de mensaje no válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2442
(2442, X'098A') El nombre de propiedad no es válido.

RC2471
(2471, X'09A7') Propiedad no disponible.

RC2469
(2469, X'09A5') El valor BufferLength es demasiado pequeño para contener las propiedades especificadas.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQMHBUF(HCONN : HMSG : MHBOPT :
                                        PRNAME : MSGDSC : BUFLEN :
                                        BUFFER : DATLEN :
                                        CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Abrir objeto) en IBM i

La llamada MQOPEN establece el acceso a un objeto.

Son válidos los siguientes tipos de objeto:

- Cola (incluidas las listas de distribución)
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Índice

- “Sintaxis” en la página 1354

- “Notas de uso” en la página 1354
- “Parámetros” en la página 1358
- “Declaración RPG” en la página 1365

Sintaxis

MQOPEN *(HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON)*

Notas de uso

1. El objeto abierto es uno de los siguientes:
   - Una cola, para:
     - Obtener o examinar mensajes (utilizando la llamada MQGET)
     - Transferir mensajes (utilizando la llamada MQPUT)
     - Consultar los atributos de la cola (utilizando la llamada MQINQ)
     - Establecer los atributos de la cola (utilizando la llamada MQSET)

     Si la cola especificada es una cola modelo, se crea una cola local dinámica.

     Una lista de distribución es un tipo especial de objeto de cola que contiene una lista de colas. Se puede abrir para transferir mensajes, pero no para obtener o examinar mensajes, ni para consultar o establecer atributos. Consulte la nota de uso 8 para obtener más detalles.

     Una cola que tiene `QSGDISP(GROUP)` es un tipo especial de definición de cola que no se puede utilizar con las llamadas MQOPEN o MQPUT1 .
   - Una lista de nombres, para:
     - Consultar los nombres de las colas de la lista (utilizando la llamada MQINQ).
   - Una definición de proceso, para:
     - Consultar sobre los atributos de proceso (utilizando la llamada MQINQ).
   - El gestor de colas, para:
     - Consultar los atributos del gestor de colas local (utilizando la llamada MQINQ).
2. Es válido que una aplicación abra el mismo objeto más de una vez. Se devuelve un descriptor de objeto diferente para cada apertura. Cada descriptor de contexto que se devuelve se puede utilizar para las funciones para las que se ha realizado la apertura correspondiente.
3. Si el objeto que se está abriendo es una cola pero no una cola de clúster, toda la resolución de nombres dentro del gestor de colas local tiene lugar en el momento de la llamada MQOPEN. Esto puede incluir uno o más de los siguientes para una llamada MQOPEN determinada:
   - Resolución de alias para el nombre de una cola base
   - Resolución del nombre de una definición local de una cola remota al nombre del gestor de colas remoto y el nombre por el que se conoce la cola en el gestor de colas remoto
   - Resolución del nombre del gestor de colas remoto en el nombre de una cola de transmisión local

   Sin embargo, tenga en cuenta que las llamadas MQINQ o MQSET posteriores para el descriptor de contexto se relacionan únicamente con el nombre que se ha abierto y no con el objeto resultante después de que se haya producido la resolución de nombres. Por ejemplo, si el objeto abierto es un alias, los atributos devueltos por la llamada MQINQ son los atributos del alias, no los atributos de la cola base en la que se resuelve el alias. Sin embargo, la comprobación de resolución de nombres se sigue realizando, independientemente de lo que se especifique para el parámetro **OPTS** en el MQOPEN correspondiente.

   Si el objeto que se está abriendo es una cola de clúster, la resolución de nombres se puede producir en el momento de la llamada MQOPEN, o se puede aplazar hasta más adelante. El punto en el

que se produce la resolución está controlado por las opciones OOBND* especificadas en la llamada MQOPEN:

- OOBNDO
- OOBNDN
- OOBNDQ

Consulte Resolución de nombres para obtener más información sobre la resolución de nombres para colas de clúster.

4. Los atributos de un objeto pueden cambiar mientras una aplicación tiene el objeto abierto. En muchos casos, la aplicación no se da cuenta de ello, pero para determinados atributos el gestor de colas marca el descriptor de contexto como ya no válido. Son las siguientes:
   - Cualquier atributo que afecte a la resolución de nombres del objeto. Esto se aplica independientemente de las opciones de apertura utilizadas e incluye lo siguiente:
     - Un cambio en el atributo **BaseQName** de una cola alias que está abierta.
     - Un cambio en los atributos de cola **RemoteQName** o **RemoteQMgrName** , para cualquier descriptor de contexto que esté abierto para esta cola, o para una cola que se resuelva a través de esta definición como un alias de gestor de colas.
     - Cualquier cambio que haga que un descriptor de contexto abierto actualmente para una cola remota se resuelva en una cola de *transmisión* diferente, o que no se pueda resolver en una en absoluto. Por ejemplo, esto puede incluir:
       - Un cambio en el atributo **XmitQName** de la definición local de una cola remota, si la definición se está utilizando para una cola o para un alias de gestor de colas.

       Hay una excepción a esto, a saber, la creación de una nueva cola de transmisión. Un descriptor de contexto que se habría resuelto en esta cola si estuviera presente cuando se abrió el descriptor de contexto, pero que en su lugar se hubiera resuelto en la cola de transmisión predeterminada, no se ha convertido en no válido.
     - Un cambio en el atributo del gestor de colas **DefXmitQName** . En este caso, todos los descriptores de contexto abiertos que se han resuelto en la cola especificada anteriormente (que se ha resuelto sólo porque era la cola de transmisión predeterminada) se marcan como no válidos. Los manejadores que se han resuelto en esta cola por otras razones no se ven afectados.
   - El atributo de cola **Shareability** , si hay dos o más descriptores de contexto que proporcionan actualmente acceso OOINPS para esta cola, o para una cola que se resuelve en esta cola. Si es así, *todos* los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, se marcan como no válidos, independientemente de las opciones de apertura.
   - El atributo de cola **Usage** , para todos los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, independientemente de las opciones de apertura.

   Cuando un descriptor de contexto se marca como no válido, todas las llamadas posteriores (distintas de MQCLOSE) que utilizan este descriptor de contexto fallan con el código de razón RC2041; la aplicación debe emitir una llamada MQCLOSE (utilizando el descriptor de contexto original) y, a continuación, volver a abrir la cola. Las actualizaciones no confirmadas en el descriptor de contexto antiguo de las llamadas satisfactorias anteriores se pueden seguir confirmando o restituyendo, según lo requiera la lógica de la aplicación.

   Si el cambio de un atributo hace que esto suceda, se debe utilizar una versión especial "force" del mandato.
5. El gestor de colas realiza comprobaciones de seguridad cuando se emite una llamada MQOPEN, para verificar que el identificador de usuario bajo el que se ejecuta la aplicación tiene el nivel de autorización adecuado antes de que se permita el acceso. La comprobación de autorización se realiza en el nombre del objeto que se está abriendo, y no en el nombre o nombres, lo que resulta después de que se haya resuelto un nombre.

Si el objeto que se está abriendo es una cola modelo, el gestor de colas realiza una comprobación de seguridad completa con el nombre de la cola modelo y el nombre de la cola dinámica que se crea. Si la cola dinámica resultante se abre entonces de forma explícita, se realiza una comprobación de seguridad de recursos adicional con respecto al nombre de la cola dinámica.

6. Una cola remota se puede especificar de una de dos maneras en el parámetro **OBJDSC** de esta llamada (consulte los campos *ODON* y *ODMN* descritos en "MQOD (Descriptor de objeto) en IBM i" en la página 1186):

   - Especificando para *ODON* el nombre de una definición local de la cola remota. En este caso, *ODMN* hace referencia al gestor de colas local y se puede especificar como espacios en blanco.

     La validación de seguridad realizada por el gestor de colas local verifica que el usuario tiene autorización para abrir la definición local de la cola remota.
   - Especificando para *ODON* el nombre de la cola remota tal como la conoce el gestor de colas remoto. En este caso, *ODMN* es el nombre del gestor de colas remoto.

     La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado a enviar mensajes a la cola de transmisión como resultado del proceso de resolución de nombres.

   En cualquier caso:

   - El gestor de colas local no envía mensajes al gestor de colas remoto para comprobar que el usuario está autorizado a colocar mensajes en la cola.
   - Cuando llega un mensaje al gestor de colas remoto, el gestor de colas remoto puede rechazarlo porque el usuario que ha originado el mensaje no está autorizado.
7. Una llamada MQOPEN con la opción OOBRW establece un cursor de examen, para su uso con llamadas MQGET que especifican el descriptor de objeto y una de las opciones de examen. Esto permite explorar la cola sin alterar su contenido. Un mensaje que se ha encontrado al examinar se puede eliminar posteriormente de la cola utilizando la opción GMMUC.

   Varios cursores de examen pueden estar activos para una sola aplicación emitiendo varias solicitudes MQOPEN para la misma cola.
8. Las siguientes notas se aplican al uso de listas de distribución.

   - Los campos de la estructura MQOD deben establecerse de la forma siguiente al abrir una lista de distribución:
     - *ODVER* debe ser ODVER2 o superior.
     - *ODOT* debe ser OTQ.
     - *ODON* debe estar en blanco o la serie nula.
     - *ODMN* debe estar en blanco o la serie nula.
     - *ODREC* Tiene que ser mayor que cero.
     - Uno de *ODORO* y *ODORP* debe ser cero y el otro distinto de cero.
     - No más de uno de *ODRRO* y *ODRRP* puede ser distinto de cero.
     - Debe haber *ODREC* registros de objeto, direccionado por *ODORO* o *ODORP*. Los registros de objeto deben establecerse en los nombres de las colas de destino que se van a abrir.
     - Si uno de *ODRRO* y *ODRRP* es distinto de cero, debe haber *ODREC* registros de respuesta presentes. Los establece el gestor de colas si la llamada se completa con el código de razón RC2136.

     Un MQOD version-2 también se puede utilizar para abrir una única cola que no esté en una lista de distribución, asegurándose de que *ODREC* sea cero.
   - Sólo las siguientes opciones de apertura son válidas en el parámetro **OPTS** :
     - OOOUT
     - OOPA*

  - OOSET*
  - OOALTU
  - OOFIQ
- Las colas de destino de la lista de distribución pueden ser colas locales, alias o remotas, pero no pueden ser colas modelo. Si se especifica una cola modelo, dicha cola no se puede abrir, con el código de razón RC2057. Sin embargo, esto no impide que otras colas de la lista se abran correctamente.
- Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:
  - Si las operaciones abiertas para las colas de la lista de distribución son satisfactorias o fallan de la misma forma, los parámetros de código de terminación y código de razón se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

    Por ejemplo, si cada apertura se realiza correctamente, el código de terminación se establece en CCOK y el código de razón es RCNONE; si cada apertura falla porque no existe ninguna de las colas, los parámetros se establecen en CCFAIL y RC2085.
  - Si las operaciones abiertas para las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
    - El parámetro de código de terminación se establece en CCWARN si al menos una apertura se ha realizado correctamente, y en CCFAIL si todo ha fallado.
    - El parámetro de código de razón se establece en RC2136.
    - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.
- Cuando una lista de distribución se ha abierto correctamente, el descriptor de contexto *HOBJ* devuelto por la llamada se puede utilizar en llamadas MQPUT posteriores para colocar mensajes en las colas de la lista de distribución y en una llamada MQCLOSE para renunciar al acceso a la lista de distribución. La única opción de cierre válida para una lista de distribución es CONONE.

  La llamada MQPUT1 también se puede utilizar para colocar un mensaje en una lista de distribución; la estructura MQOD que define las colas de la lista se especifica como un parámetro en dicha llamada.
- Cada destino abierto correctamente en la lista de distribución cuenta como un descriptor de contexto *separado* al comprobar si la aplicación ha superado el número máximo permitido de descriptores de contexto (consulte el atributo de gestor de colas **MaxHandles**). Esto es cierto incluso cuando dos o más de los destinos de la lista de distribución se resuelven realmente en la misma cola física. Si la llamada MQOPEN o MQPUT1 para una lista de distribución haría que el número de descriptores de contexto utilizados por la aplicación excediera de *MaxHandles*, la llamada falla con el código de razón RC2017.
- Cada destino que se abre correctamente tiene el valor de su atributo **OpenOutputCount** incrementado en uno. Si dos o más de los destinos de la lista de distribución se resuelven realmente en la misma cola física, el atributo **OpenOutputCount** de dicha cola se incrementa en el número de destinos de la lista de distribución que se resuelven en dicha cola.
- Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.
- Es válido que una lista de distribución contenga sólo un destino.

9. Las notas siguientes se aplican al uso de colas de clúster.
   - Cuando una cola de clúster se abre por primera vez, y el gestor de colas local no es un gestor de colas de depósito completo, el gestor de colas local obtiene información sobre la cola de clúster de un gestor de colas de depósito completo. Cuando la red está ocupada, el gestor de colas local puede tardar varios segundos en recibir la información necesaria del gestor de colas de repositorio.

Como resultado, es posible que la aplicación que emite la llamada MQOPEN tenga que esperar hasta 10 segundos antes de que se devuelva el control de la llamada MQOPEN. Si el gestor de colas local no recibe la información necesaria sobre la cola de clúster dentro de este tiempo, la llamada falla con el código de razón RC2189.

- Cuando se abre una cola de clúster y hay varias instancias de la cola en el clúster, la instancia realmente abierta depende de las opciones especificadas en la llamada MQOPEN:
  - Si las opciones especificadas incluyen alguna de las siguientes:
    - OOBRW
    - OOINPQ
    - OOINPX
    - OOINPS
    - OOSET

    la instancia de la cola de clúster abierta debe ser la instancia local. Si no hay ninguna instancia local de la cola, la llamada MQOPEN falla.
  - Si las opciones especificadas no incluyen ninguna de las opciones anteriores, pero sí incluyen una o ambas de las siguientes:
    - OOINQ
    - OOOUT

    la instancia abierta es la instancia local si hay una y, de lo contrario, una instancia remota. Sin embargo, la instancia elegida por el gestor de colas puede ser alterada por una salida de carga de trabajo de clúster (si la hay).

Para obtener más información sobre las colas de clúster, consulte Colas de clúster.

10. A las aplicaciones iniciadas por un supervisor desencadenante se les pasa el nombre de la cola asociada con la aplicación cuando se inicia la aplicación. Este nombre de cola se puede especificar en el parámetro **OBJDSC** para abrir la cola. Consulte la descripción de la estructura MQTMC para obtener más detalles.
11. Cuando se utiliza la opción OORLOQ, la cola local ya se devuelve cuando se abre una cola local, alias o modelo, pero este no es el caso cuando, por ejemplo, se abre una cola remota o una cola de clúster no local; el nombre ResolvedQName y ResolvedQMgrse especifican con el nombre RemoteQName y RemoteQMgrque se encuentra en la definición de cola remota, o de forma similar con la cola de clúster remoto elegida. Si se especifica OORLOQ al abrir, por ejemplo, una cola remota, ResolvedQName será ahora la cola de transmisión a la que se colocarán los mensajes. El nombre ResolvedQMgrse especificará con el nombre del gestor de colas local que aloja la cola de transmisión. Si un usuario tiene autorización para examinar, entrar o salir en una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQOPEN. No se necesita ninguna autorización especial.

Parámetros

La llamada MQOPEN tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

OBJDSC (MQOD)-entrada/salida

Descriptor de objeto.

Esta es una estructura que identifica el objeto que se va a abrir; consulte “MQOD (Descriptor de objeto) en IBM i” en la página 1186 para obtener más detalles.

Si el campo *ODON* del parámetro **OBJDSC** es el nombre de una cola modelo, una cola local dinámica se crea con los atributos de la cola modelo; esto sucede independientemente de las opciones de apertura especificadas por el parámetro **OPTS** . Las operaciones posteriores que utilizan *HOBJ* devueltas por la llamada MQOPEN se realizan en la nueva cola dinámica y no en la cola modelo. Esto es cierto incluso para las llamadas MQINQ y MQSET. El nombre de la cola modelo en el parámetro **OBJDSC** se sustituye por el nombre de la cola dinámica creada. El tipo de la cola dinámica viene determinado por el valor del atributo **DefinitionType** de la cola modelo (consulte "Atributos para colas" en la página 1402). Para obtener información sobre las opciones de cierre aplicables a las colas dinámicas, consulte la descripción de la llamada MQCLOSE.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQOPEN.

Se debe especificar al menos una de las opciones siguientes:

- OOBRW
- OOINP* (sólo uno de ellos)
- OOINQ
- OOOUT
- OOSET
- OORLQ

Se pueden especificar otras opciones según sea necesario. Si se necesita más de una opción, se pueden añadir los valores (no añadir la misma constante más de una vez). Las combinaciones que no son válidas se anotan; todas las demás combinaciones son válidas. Sólo se permiten las opciones que son aplicables al tipo de objeto especificado por *OBJDSC* (consulte Opciones MQOPEN válidas para cada tipo de cola).

Opciones de acceso: las opciones siguientes controlan el tipo de operaciones que se pueden realizar en el objeto:

OOINPQ
Abra la cola para obtener mensajes utilizando el valor predeterminado definido por la cola.

La cola se abre para su uso con las llamadas MQGET posteriores. El tipo de acceso es compartido o exclusivo, en función del valor del atributo de cola **DefInputOpenOption** ; consulte "Atributos para colas" en la página 1402 para obtener más detalles.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

OOINPS
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con OOINPS, pero falla con el código de razón RC2042 si la cola está abierta actualmente con OOINPX.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

OOINPX
Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón RC2042 si la cola está abierta actualmente por esta u otra aplicación para entrada de cualquier tipo (OOINPS u OOINPX).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

Las siguientes notas se aplican a estas opciones:

- Sólo se puede especificar una de estas opciones.
- Una llamada MQOPEN con una de estas opciones puede ser satisfactoria incluso si el atributo de cola **InhibitGet** se establece en QAGETI (aunque las llamadas MQGET posteriores fallarán mientras el atributo se establece en este valor).
- Si la cola está definida como no compartible (es decir, el atributo de cola **Shareability** tiene el valor QANSHR), los intentos de abrir la cola para acceso compartido se tratan como intentos de abrir la cola con acceso exclusivo.
- Si se abre una cola alias con una de estas opciones, la prueba de uso exclusivo (o si otra aplicación tiene uso exclusivo) se realiza en la cola base en la que se resuelve el alias.
- Estas opciones no son válidas si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

OOBRW

Abra la cola para examinar los mensajes.

La cola se abre para su uso con llamadas MQGET posteriores con una de las opciones siguientes:

- GMBRWF
- GMBRWN
- GMBRWC

Esto se permite incluso si la cola está abierta actualmente para OOINPX. Una llamada MQOPEN con la opción OOBRW establece un cursor de examen y lo sitúa lógicamente antes del primer mensaje en la cola; consulte el campo *GMOPT* descrito en "MQGMO (Opciones de obtención de mensajes) en IBM i" en la página 1098 para obtener más información.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Tampoco es válido si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

OOOUT

Abra la cola para transferir mensajes, o un tema o serie de tema para publicar mensajes.

La cola se abre para su uso con llamadas MQPUT posteriores.

Una llamada MQOPEN con esta opción puede ser satisfactoria incluso si el atributo de cola **InhibitPut** se establece en QAPUTI (aunque las llamadas MQPUT posteriores fallarán mientras el atributo se establezca en este valor).

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución y los temas.

OOINQ

Abrir objeto para consultar atributos.

La cola, la lista de nombres, la definición de proceso o el gestor de colas se abre para su uso con llamadas MQINQ posteriores.

Esta opción es válida para todos los tipos de objetos que no sean listas de distribución. No es válido si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para la asignación de alias de gestor de colas es el nombre del gestor de colas local.

OOSET

Abra la cola para establecer atributos.

La cola se abre para utilizarla con las llamadas MQSET posteriores.

Esta opción es válida para todos los tipos de cola que no sean listas de distribución. No es válido si *ODMN* es el nombre de una definición local de una cola remota; esto es cierto incluso si el valor

del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

Opciones de enlace: Las opciones siguientes se aplican cuando el objeto que se está abriendo es una cola de clúster; estas opciones controlan el enlace del descriptor de contexto de cola a una instancia de la cola de clúster:

OOBNDO
Manejador de enlace a destino cuando se abre la cola.

Esto hace que el gestor de colas local enlace el descriptor de contexto de cola a una instancia de la cola de destino cuando se abre la cola. Como resultado, todos los mensajes colocados utilizando este descriptor de contexto se envían a la misma instancia de la cola de destino y por la misma ruta.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

OOBNDN
No enlazar a un destino específico.

Esto detiene el gestor de colas local que enlaza el descriptor de contexto de cola con una instancia de la cola de destino. Como resultado, las llamadas MQPUT sucesivas que utilizan este descriptor de contexto pueden hacer que los mensajes se envíen a *diferentes* instancias de la cola de destino, o que se envíen a la misma instancia pero mediante rutas diferentes. También permite que el gestor de colas local, un gestor de colas remoto o un agente de canal de mensajes (MCA) cambien la instancia seleccionada más tarde, según las condiciones de red.

Nota: Las aplicaciones cliente y servidor que necesitan intercambiar una *serie* de mensajes para completar una transacción no deben utilizar OOBNDN (u OOBNDQ cuando *DefBind* tiene el valor BNDNOT), porque los mensajes sucesivos de la serie se pueden enviar a diferentes instancias de la aplicación de servidor.

Si se especifica OOBRW o una de las opciones OOINP* para una cola de clúster, el gestor de colas se fuerza a seleccionar la instancia local de la cola de clúster. Como resultado, el enlace del descriptor de contexto de cola es fijo, incluso si se especifica OOBNDN.

Si se especifica OOINQ con OOBNDN, las llamadas MQINQ sucesivas que utilizan ese manejador pueden consultar distintas instancias de la cola de clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

OOBNDN sólo es válido para las colas y sólo afecta a las colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

OOBNDQ
Utilice el enlace predeterminado para la cola.

Esto hace que el gestor de colas local enlace el descriptor de contexto de cola de la forma definida por el atributo de cola **DefBind** . El valor de este atributo es BNDOPN o BNDNOT.

OOBNDQ es el valor por omisión si no se especifican OOBNDO y OOBNDN.

OOBNDQ se define para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquiera de las otras dos opciones de vinculación, pero debido a que su valor es cero, no se puede detectar dicho uso.

Opciones de contexto: Las opciones siguientes controlan el proceso del contexto de mensaje:

OOSAVA
Guardar contexto cuando se recupere el mensaje.

La información de contexto está asociada con este descriptor de contexto de cola. Esta información se establece a partir del contexto de cualquier mensaje recuperado utilizando este descriptor de contexto. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta información de contexto se puede pasar a un mensaje que se coloca posteriormente en una cola utilizando las llamadas MQPUT o MQPUT1 . Consulte las opciones PMPASI y PMPASA descritas en “MQPMO (opciones de transferencia de mensajes) en IBM i” en la página 1200.

Hasta que un mensaje se haya recuperado correctamente, el contexto no se puede pasar a un mensaje que se está colocando en una cola.

Un mensaje recuperado utilizando una de las opciones de examen GMBRW* no tiene guardada su información de contexto (aunque los campos de contexto del parámetro **MSGDSC** se establecen después de un examen).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Debe especificarse una de las opciones OOINP*.

OOPASI
Permitir que se pase el contexto de identidad.

Esto permite especificar la opción PMPASI en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad de una cola de entrada que se ha abierto con la opción OOSAVA. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOPASA
Permitir que se pase todo el contexto.

Esto permite especificar la opción PMPASA en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen de una cola de entrada que se ha abierto con la opción OOSAVA. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica OOPASI, que por lo tanto no es necesario especificar. Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOOSETI
Permitir que se establezca el contexto de identidad.

Esto permite especificar la opción PMSETI en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad contenida en el parámetro **MSGDSC** especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica OOPASI, que por lo tanto no es necesario especificar. Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOSETA
Permitir que se establezca todo el contexto.

Esto permite especificar la opción PMSETA en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen contenida en el parámetro **MSGDSC** especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta opción implica las siguientes opciones, por lo que no es necesario especificarlas:

- OOPASI
- OOPASA

- OOOSETI

  Debe especificarse la opción OOOUT.

  Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

Otras opciones: Las opciones siguientes controlan la comprobación de autorización y lo que sucede cuando el gestor de colas se desactiva temporalmente:

OOALTU
Validar con el identificador de usuario especificado.

Esto indica que el campo *ODAU* del parámetro **OBJDSC** contiene un identificador de usuario que se va a utilizar para validar esta llamada MQOPEN. La llamada sólo puede realizarse correctamente si este *ODAU* está autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo. Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.

Esta opción es válida para todos los tipos de objeto.

OOFIQ
Fallar si el gestor de colas se está desactivando temporalmente.

Esta opción fuerza a que la llamada MQOPEN falle si el gestor de colas está en estado de desactivación temporal.

Esta opción es válida para todos los tipos de objeto.

OORLQ
Especifique el nombre de la cola local que se ha abierto.

Esta opción especifica que el ResolvedQName de la estructura MQOD (si está disponible) debe especificarse con el nombre de la cola local que se ha abierto. El nombre ResolvedQMgrse especificará de forma similar con el nombre del gestor de colas local que aloja la cola local.

Tabla 750. Opciones MQOPEN válidas para cada tipo de cola

Opción	Alias (“1” en la página 1364)	Local y modelo	Remotos	Clúster no local	Lista de distribución	Tema
OOINPQ	✓	✓	-	-	-	-
OOINPS	✓	✓	-	-	-	-
OOINPX	✓	✓	-	-	-	-
OOBRW	✓	✓	-	-	-	-
OOOUT	✓	✓	✓	✓	✓	✓
OOINQ	✓	✓	“2” en la página 1364	✓	-	-
OOSET	✓	✓	“2” en la página 1364	-	-	-
OOBNDO (“3” en la página 1364)	✓	✓	✓	✓	✓	-
OOBNDN (“3” en la página 1364)	✓	✓	✓	✓	✓	-

Tabla 750. Opciones MQOPEN válidas para cada tipo de cola (continuación)						
Opción	**Alias ("1" en la página 1364)**	**Local y modelo**	**Remotos**	**Clúster no local**	**Lista de distribución**	**Tema**
OOBNDQ ("3" en la página 1364)	✓	✓	✓	✓	✓	-
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	"5" en la página 1364
OOPASA	✓	✓	✓	✓	✓	"5" en la página 1364
OOOSETI	✓	✓	✓	✓	✓	"5" en la página 1364
OOSETA	✓	✓	✓	✓	✓	"5" en la página 1364
OOALTU	✓	✓	✓	✓	✓	✓
OOFIQ	✓	✓	✓	✓	✓	✓
OORLQ	✓	✓	✓	✓	-	-

Notas:

1. La validez de las opciones para los alias depende de la validez de la opción para la cola en la que se resuelve el alias.
2. Esta opción sólo es válida para la definición local de una cola remota.
3. Esta opción se puede especificar para cualquier tipo de cola, pero se ignora si la cola no es una cola de clúster.
4. Este atributo se ignora para un tema.
5. Estos atributos se pueden utilizar con un tema, pero sólo afectan al contexto establecido para el mensaje retenido, no a los campos de contexto enviados a cualquier suscriptor.

HOBJ (entero con signo de 10 dígitos)-salida

El manejador del objeto.

Este descriptor de contexto representa el acceso que se ha establecido al objeto. Debe especificarse en las llamadas de cola de mensajes posteriores que operan en el objeto. Deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto está restringido a la unidad más pequeña de proceso paralelo soportado por la plataforma en la que se ejecuta la aplicación; el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada MQOPEN:

- En IBM i, el ámbito del descriptor de contexto es el trabajo que emite la llamada.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQOPEN(HCONN : OBJDSC : OPTS :
C                                      HOBJ : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (transferir mensaje) en IBM i

La llamada MQPUT coloca un mensaje en una cola, lista de distribución o en un tema. La cola, la lista de distribución o el tema ya deben estar abiertos.

- “Sintaxis” en la página 1365
- “Notas de uso” en la página 1365
  - “Temas” en la página 1366
  - “MQPUT y MQPUT1” en la página 1366
  - “Colas de destino” en la página 1366
  - “Listas de distribución” en la página 1367
  - “Cabeceras” en la página 1369
  - “Almacenamiento intermedio” en la página 1369
- “Parámetros” en la página 1370
- “Declaración RPG” en la página 1375

Sintaxis

MQPUT *(HCONN, HOBJ, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Notas de uso

Temas

Las notas siguientes se aplican al uso de temas:

1. Cuando se utiliza MQPUT para publicar mensajes sobre un tema, donde uno o más suscriptores de ese tema no pueden recibir la publicación debido a un problema con su cola de suscriptores (por ejemplo, está llena), el código de razón devuelto a la llamada MQPUT y el comportamiento de entrega depende del valor de los atributos PMSGDLV o NPMSGDLV en el TOPIC. Tenga en cuenta que la entrega de una publicación a la cola de mensajes no entregados cuando se especifica RODLQ, o descartar el mensaje cuando se especifica RODISC, se considera una entrega satisfactoria del mensaje. Si no se entrega ninguna de las publicaciones, MQPUT volverá con RC2502. Esto puede ocurrir en los siguientes casos:
   - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALL y cualquier suscripción (duradera o no) tiene una cola que no puede recibir la publicación.
   - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción duradera tiene una cola que no puede recibir la publicación.

   MQPUT puede volver con RCNONE aunque las publicaciones no se hayan podido entregar a algunos suscriptores en los casos siguientes:
   - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLAVAIL y cualquier suscripción, duradera o no, tiene una cola que no puede recibir la publicación.
   - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (en función de la persistencia del mensaje) establecido en ALLDUR y una suscripción no duradera tiene una cola que no puede recibir la publicación.
2. Si no hay suscriptores para el tema que se está utilizando, el mensaje publicado no se envía a ninguna cola y se descarta. No hace ninguna diferencia si este mensaje es persistente o no persistente, o si tiene una caducidad ilimitada o algún tiempo de caducidad pequeño, todavía se descarta si no hay suscriptores. La excepción a esto es si el mensaje se va a retener, en cuyo caso, aunque no se envía a las colas de ningún suscriptor, se almacena en el tema que se va a entregar a cualquier suscripción nueva o a cualquier suscriptor que solicite publicaciones retenidas utilizando MQSUBRQ.

MQPUT y MQPUT1

Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias

- La llamada MQPUT debe utilizarse cuando se van a colocar varios mensajes en la *misma* cola.

  En primer lugar se emite una llamada MQOPEN que especifica la opción OOOUT, seguida de una o varias solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .
- La llamada MQPUT1 debe utilizarse cuando sólo se va a colocar *un* mensaje en una cola.

  Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

Colas de destino

Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de estos mensajes se conserva si se cumplen las condiciones siguientes. Algunas condiciones se aplican tanto a las colas de destino locales como a las remotas; otras condiciones se aplican sólo a las colas de destino remotas.

Condiciones para colas de destino locales y remotas

- Todas las llamadas MQPUT están dentro de la misma unidad de trabajo, o ninguna de ellas está dentro de una unidad de trabajo.

Cuando los mensajes se colocan en una cola determinada dentro de una sola unidad de trabajo, los mensajes de otras aplicaciones pueden intercalarse con la secuencia de mensajes de la cola.

- Todas las llamadas MQPUT se realizan utilizando el mismo descriptor de objeto *HOBJ*.

  En algunos entornos, la secuencia de mensajes también se conserva cuando se utilizan distintos manejadores de objetos, siempre que las llamadas se realicen desde la misma aplicación. El significado de "misma aplicación" viene determinado por el entorno:

  – En IBM i, la aplicación es el trabajo.
- Los mensajes tienen todos la misma prioridad.

Condiciones adicionales para colas de destino remotas

- Sólo hay una vía de acceso desde el gestor de colas emisor al gestor de colas de destino.

  Si existe la posibilidad de que algunos mensajes de la secuencia vayan a una vía de acceso diferente (por ejemplo, debido a la reconfiguración, el equilibrio de tráfico o la selección de vía de acceso en función del tamaño del mensaje), no se puede garantizar el orden de los mensajes en el gestor de colas de destino.
- Los mensajes no se colocan temporalmente en colas de mensajes no entregados en los gestores de colas de envío, intermedios o de destino.

  Si uno o varios de los mensajes se colocan temporalmente en una cola de mensajes no entregados (por ejemplo, porque una cola de transmisión o la cola de destino está llena temporalmente), los mensajes pueden llegar a la cola de destino fuera de secuencia.
- Los mensajes son todos persistentes o todos no persistentes.

  Si un canal de la ruta entre los gestores de colas de envío y de destino tiene su atributo **CDNPM** establecido en NPFAST, los mensajes no persistentes pueden saltar por delante de los mensajes persistentes, lo que da como resultado que el orden de los mensajes persistentes relativos a los mensajes no persistentes no se conserve. Sin embargo, se conserva el orden de los mensajes persistentes relativos entre sí y de los mensajes no persistentes relativos entre sí.

Si no se cumplen estas condiciones, se pueden utilizar grupos de mensajes para conservar el orden de los mensajes, pero tenga en cuenta que esto requiere que las aplicaciones emisora y receptora utilicen el soporte de agrupación de mensajes. Para obtener más información sobre los grupos de mensajes, consulte:

- *MDMFL* Campo de MQMD
- Opción PMLOGO en MQPMO
- Opción GMLOGO en MQGMO

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

1. Los mensajes se pueden transferir a una lista de distribución utilizando version-1 o version-2 MQPMO. Si se utiliza una MQPMO version-1 (o una MQPMO version-2 con *PMREC* igual a cero), la aplicación no puede proporcionar registros de mensajes de colocación ni registros de respuesta. Esto significa que no será posible identificar las colas que encuentran errores, si el mensaje se envía correctamente a algunas colas de la lista de distribución y no a otras.

   Si la aplicación proporciona registros de mensajes de colocación o registros de respuesta, el campo *PMVER* debe establecerse en PMVER2.

   Un MQPMO version-2 también se puede utilizar para enviar mensajes a una sola cola que no esté en una lista de distribución, asegurándose de que *PMREC* sea cero.
2. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:

   - Si todas las colocaciones en las colas de la lista de distribución son satisfactorias o fallan de la misma forma, el código de terminación y los parámetros de código de razón se establecen para

describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si cada colocación es satisfactoria, el código de terminación se establece en CCOK y el código de razón es RCNONE; si cada colocación falla porque todas las colas están inhibidas para las colocaciones, los parámetros se establecen en CCFAIL y RC2051.

- Si las colocaciones en las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
  - El parámetro de código de terminación se establece en CCWARN si al menos una operación put se ha realizado correctamente, y en CCFAIL si todo ha fallado.
  - El parámetro de código de razón se establece en RC2136.
  - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

  Si la colocación en un destino falla porque la apertura para dicho destino ha fallado, los campos del registro de respuesta se establecen en CCFAIL y RC2137; dicho destino se incluye en *PMIDC*.

3. Si un destino de la lista de distribución se resuelve en una cola local, el mensaje se coloca en esa cola en formato normal (es decir, no como un mensaje de lista de distribución). Si más de un destino se resuelve en la misma cola local, se coloca un mensaje en la cola para cada destino.

   Si un destino de la lista de distribución se resuelve en una cola remota, se coloca un mensaje en la cola de transmisión adecuada. Cuando varios destinos se resuelvan en la misma cola de transmisión, podrá colocarse en la cola de transmisión un único mensaje de lista de distribución que contenga dichos destinos, incluso si dichos destinos no estuvieran adyacentes en la lista de destinos proporcionada por la aplicación. Sin embargo, esto sólo se puede hacer si la cola de transmisión da soporte a mensajes de lista de distribución (consulte el atributo de cola **DistLists** descrito en "Atributos para colas" en la página 1402).

   Si la cola de transmisión no soporta listas de distribución, se coloca una copia del mensaje en formato normal en la cola de transmisión para cada destino que utiliza dicha cola de transmisión.

   Si una lista de distribución con los datos de mensaje de aplicación es demasiado grande para una cola de transmisión, el mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños, cada uno de los cuales contiene menos destinos. Si los datos del mensaje de aplicación solo se ajustan a la cola, los mensajes de lista de distribución no se pueden utilizar en absoluto, y el gestor de colas genera una copia del mensaje en formato normal para cada destino que utiliza dicha cola de transmisión.

   Si destinos diferentes tienen una prioridad de mensaje o persistencia de mensaje diferentes (esto puede ocurrir cuando la aplicación especifica PRQDEF o PEQDEF), los mensajes no se conservan en el mismo mensaje de lista de distribución. En su lugar, el gestor de colas genera tantos mensajes de lista de distribución como sean necesarios para acomodar los diferentes valores de prioridad y persistencia.

4. Una colocación en una lista de distribución puede dar como resultado:
   - Un único mensaje de lista de distribución, o
   - Un número de mensajes de lista de distribución más pequeños, o
   - Una combinación de mensajes de lista de distribución y mensajes normales, o
   - Sólo mensajes normales.

   Cuál de las situaciones anteriores depende de si:
   - Los destinos de la lista son locales, remotos o una mezcla.
   - Los destinos tienen la misma prioridad de mensaje y persistencia de mensaje.
   - Las colas de transmisión pueden contener mensajes de lista de distribución.
   - Las longitudes máximas de mensajes de las colas de transmisión son lo suficientemente grandes para acomodar el mensaje en formato de lista de distribución.

Sin embargo, independientemente de cuál de las situaciones anteriores, cada mensaje *físico* resultante (es decir, cada mensaje normal o mensaje de lista de distribución resultante de la operación de transferir) cuenta como un solo mensaje *uno* cuando:

- Comprobación de si la aplicación ha superado el número máximo permitido de mensajes en una unidad de trabajo (consulte el atributo del gestor de colas **MaxUncommittedMsgs**).
- Comprobando si se cumplen las condiciones de desencadenamiento.
- Incrementando las profundidades de cola y comprobando si se superaría la profundidad máxima de cola de las colas.

5. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.

Cabeceras

Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Si el gestor de colas detecta un error, la llamada falla con un código de razón adecuado. Las comprobaciones realizadas varían según las estructuras particulares que estén presentes. Además, las comprobaciones sólo se realizan si se utiliza un MQMD version-2 o posterior en la llamada MQPUT o MQPUT1 ; las comprobaciones no se realizan si se utiliza un MQMD version-1 , incluso si hay un MQMDE al principio de los datos del mensaje de aplicación.

El gestor de colas valida completamente las siguientes estructuras de cabecera IBM MQ : MQDH, MQMDE.

Para otras estructuras de cabecera de IBM MQ , el gestor de colas realiza alguna validación, pero no comprueba cada campo. Las estructuras que no están soportadas por el gestor de colas local y las estructuras que siguen a la primera MQDLH del mensaje no se validan.

Además de las comprobaciones generales en los campos de las estructuras IBM MQ , deben cumplirse las condiciones siguientes:

- Una estructura IBM MQ no debe dividirse en dos o más segmentos; la estructura debe estar totalmente contenida en un segmento.
- La suma de las longitudes de las estructuras de un mensaje PCF debe ser igual a la longitud especificada por el parámetro **BUFLEN** en la llamada MQPUT o MQPUT1 . Un mensaje PCF es un mensaje que tiene uno de los siguientes nombres de formato:
  - FMADMN
  - FMEVNT
  - FMPCF
- Las estructuras IBM MQ no se deben truncar, excepto en las situaciones siguientes en las que se permiten estructuras truncadas:
  - Mensajes que son mensajes de informe.
  - Mensajes PCF.
  - Mensajes que contienen una estructura MQDLH. (Las estructuras *que siguen* a la primera MQDLH se pueden truncar; las estructuras que preceden a la MQDLH no se pueden truncar.)

Almacenamiento intermedio

El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico. Esto aumentará la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQPUT tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este descriptor de contexto representa la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. El valor de *HOBJ* lo ha devuelto una llamada MQOPEN anterior que ha especificado la opción OOOUT.

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje que se envía y recibe información sobre el mensaje una vez completada la solicitud de colocación. Consulte "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener los detalles.

Si la aplicación proporciona un MQMD version-1 , los datos del mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *MDFMT* en MQMD debe establecerse en FMMDE para indicar que hay un MQMDE presente. Consulte "MQMDE (extensión de descriptor de mensaje) en IBM i" en la página 1179 para obtener más detalles.

PMO (MQPMO)-entrada/salida

Opciones que controlan la acción de MQPUT.

Consulte "MQPMO (opciones de transferencia de mensajes) en IBM i" en la página 1200 para obtener los detalles.

BUFLEN (entero con signo de 10 dígitos)-entrada

Longitud del mensaje en *BUFFER*.

Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior para *BUFLEN* depende de varios factores:

- Si la cola de destino es una cola compartida, el límite superior es de 63 KB (64 512 bytes).
- Si el destino es una cola local o se resuelve en una cola local (pero no es una cola compartida), el límite superior depende de si:
  - El gestor de colas local da soporte a la segmentación.
  - La aplicación emisora especifica el distintivo que permite al gestor de colas segmentar el mensaje. Este distintivo es MFSEGA, y se puede especificar en un MQMD de version-2 o en un MQMDE utilizado con un MQMD de version-1 .

  Si se cumplen ambas condiciones, *BUFLEN* no puede exceder de 999 999 999 menos el valor del campo *MDOFF* en MQMD. El mensaje lógico más largo que se puede transferir es, por lo tanto, 999 999 999 bytes (cuando *MDOFF* es cero). Sin embargo, las restricciones de recursos impuestas por el sistema operativo o el entorno en el que se ejecuta la aplicación pueden dar como resultado un límite inferior.

  Si una o ambas de las condiciones descritas anteriormente no se cumplen, *BUFLEN* no puede exceder el atributo **MaxMsgLength** y el atributo **MaxMsgLength** del gestor de colas.
- Si el destino es una cola remota o se resuelve en una cola remota, se aplican las condiciones para las colas locales, *pero en cada gestor de colas a través del cual debe pasar el mensaje para llegar a la cola de destino* ; en particular:

1. La cola de transmisión local utilizada para almacenar el mensaje temporalmente en el gestor de colas local
2. Colas de transmisión intermedias (si las hay) utilizadas para almacenar el mensaje en los gestores de colas de la ruta entre los gestores de colas local y de destino
3. La cola de destino en el gestor de colas de destino

Por lo tanto, el mensaje más largo que se puede transferir está gobernado por el más restrictivo de estas colas y gestores de colas.

Cuando un mensaje está en una cola de transmisión, la información adicional reside en los datos del mensaje, y esto reduce la cantidad de datos de aplicación que pueden transportarse. En esta situación, se recomienda que los bytes LNMHD se resten de los valores *MaxMsgLength* de las colas de transmisión al determinar el límite para *BUFLEN*.

Nota: Sólo el incumplimiento de la condición 1 se puede diagnosticar de forma síncrona (con el código de razón RC2030 o RC2031) cuando se coloca el mensaje. Si no se cumplen las condiciones 2 o 3, el mensaje se redirige a una cola de mensajes no entregados, ya sea en un gestor de colas intermedio o en el gestor de colas de destino. Si esto sucede, se genera un mensaje de informe si el remitente lo ha solicitado.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada

Los datos del mensaje.

Se trata de un almacenamiento intermedio que contiene los datos de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *BUFFER* contiene datos de tipo carácter, datos numéricos o ambos, los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** deben establecerse en los valores adecuados para los datos; esto permitirá al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2104
(2104, X'838 ') No se reconoce la opción de informe en el descriptor de mensaje.

RC2136
(2136, X'858 ') Se han devuelto varios códigos de razón.

Si *CMPCOD* es CCFAIL:

RC2004
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2013
(2013, X'7DD') Tiempo de caducidad no válido.

RC2014
(2014, X'7DE') Código de comentarios no válido.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2024
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2026
(2026, X'7EA') El descriptor de mensaje no es válido.

RC2027
(2027, X'7EB') Falta la cola de respuestas.

RC2029
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

RC2030
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

RC2031
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

RC2039
(2039, X'7F7') Cola no abierta para salida.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2047
(2047, X'7FF') Persistencia no válida.

RC2048
(2048, X'800 ') La cola no admite mensajes persistentes.

RC2050
(2050, X'802 ') La prioridad del mensaje no es válida.

RC2051
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2053
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

RC2056
(2056, X'808 ') No hay espacio disponible en disco para la cola.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2061
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2072
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2093
(2093, X'82D') Cola no abierta para pasar todo el contexto.

RC2094
(2094, X'82E') La cola no está abierta para el contexto de identidad de paso.

RC2095
(2095, X'82F') Cola no abierta para establecer todo el contexto.

RC2096
(2096, X'830 ') La cola no está abierta para el contexto de identidad establecido.

RC2097
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

RC2098
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2135
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

RC2136
(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2137
(2137, X'859 ') El objeto no se ha abierto correctamente.

RC2149
(2149, X'865 ') Estructuras PCF no válidas.

RC2154
(2154, X'86A') Número de registros presentes no válido.

RC2156
(2156, X'86C') Los registros de respuesta no son válidos.

RC2158
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

RC2159
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2173
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

RC2185
(2185, X'889 ') Especificación de persistencia incoherente.

RC2188
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

RC2189
(2189, X'88D') La resolución de nombres de clúster ha fallado.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2241
(2241, X'8C1') El grupo de mensajes no está completo.

RC2242
(2242, X'8C2') El mensaje lógico no está completo.

RC2245
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2248
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

RC2249
(2249, X'8C9') Los distintivos de mensaje no son válidos.

RC2250
(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2251
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

RC2252
(2252, X'8CC') Longitud original no válida.

RC2253
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

RC2255
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2257
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2258
(2258, X'8D2') Identificador de grupo no válido.

RC2266
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

RC2269
(2269, X'8DD') Error de recurso de clúster.

RC2270
(2270, X'8DE') No hay colas de destino disponibles.

RC2420
(2420) Se ha emitido una llamada MQPUT, pero los datos del mensaje contienen una estructura MQEPH que no es válida.

RC2479
(2479, X'9AF') No se ha podido retener la publicación.

RC2480
(2480, X'9B0') El tipo de destino ha cambiado: la cola alias hacía referencia a una cola pero ahora hace referencia a un tema.

RC2502
(2502, X'9C6') La publicación ha fallado y la publicación no se ha entregado a ningún suscriptor

RC2551
(2551, X'9F7') La serie de selección especificada no está disponible.

RC2554
(2554, X'9FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje debe entregarse a un suscriptor con un selector de mensajes ampliado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQPUT(HCONN : HOBJ : MSGDSC : PMO :
C                                     BUFLEN : BUFFER : CMPCOD :
C                                     REASON)
```

La definición de prototipo para la llamada es:

```
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 (Colocar un mensaje) en IBM i

La llamada MQPUT1 coloca un mensaje en una cola o lista de distribución, o en un tema. No es necesario que la cola, la lista de distribución o el tema estén abiertos.

- “Sintaxis” en la página 1375
- “Notas de uso” en la página 1375
- “Parámetros” en la página 1376
- “Declaración RPG” en la página 1381

Sintaxis

MQPUT1 *(HCONN, OBJDSC, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Notas de uso

1. Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias:
   - La llamada MQPUT debe utilizarse cuando se van a colocar varios mensajes en la *misma* cola.

En primer lugar se emite una llamada MQOPEN que especifica la opción OOOUT, seguida de una o varias solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .

- La llamada MQPUT1 debe utilizarse cuando sólo se va a colocar *un* mensaje en una cola.

  Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

2. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Sin embargo, en la mayoría de los entornos, la llamada MQPUT1 no satisface estas condiciones, por lo que no conserva el orden de los mensajes. En su lugar, se debe utilizar la llamada MQPUT en estos entornos. Consulte las notas de uso en la descripción de la llamada MQPUT para obtener detalles.
3. La llamada MQPUT1 se puede utilizar para transferir mensajes a listas de distribución. Para obtener información general sobre esto, consulte las notas de uso para las llamadas MQOPEN y MQPUT.

   Las diferencias siguientes se aplican cuando se utiliza la llamada MQPUT1 :

   a. Si la aplicación proporciona registros de respuesta MQRR, se deben proporcionar utilizando la estructura MQOD; no se pueden proporcionar utilizando la estructura MQPMO.
   b. MQPUT1 nunca devuelve el código de razón RC2137 en los registros de respuesta; si una cola no se puede abrir, el registro de respuesta para dicha cola contiene el código de razón real resultante de la operación de apertura.

      Si una operación de apertura para una cola es satisfactoria con un código de terminación de CCWARN, el código de terminación y el código de razón del registro de respuesta para dicha cola se sustituyen por los códigos de terminación y razón resultantes de la operación de transferencia.

      Al igual que con las llamadas MQOPEN y MQPUT, el gestor de colas establece los registros de respuesta (si se proporcionan) sólo cuando el resultado de la llamada no es el mismo para todas las colas de la lista de distribución; esto se indica mediante la llamada que se completa con el código de razón RC2136.
4. Si se utiliza la llamada MQPUT1 para colocar un mensaje en una cola de clúster, la llamada se comporta como si se hubiera especificado OOBNDN en la llamada MQOPEN.
5. Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Para obtener más información sobre esto, consulte las notas de uso para la llamada MQPUT.
6. Si surge más de una de las situaciones de aviso (consulte el parámetro **CMPCOD**), el código de razón devuelto es el *primero* de la lista siguiente que se aplica:

   a. RC2136
   b. RC2242
   c. RC2241
   d. RC2049 o RC2104
7. El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico. Esto aumentará la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQPUT1 tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

OBJDSC (MQOD)-entrada/salida

Descriptor de objeto.

Es una estructura que identifica la cola a la que se añade el mensaje. Consulte "MQOD (Descriptor de objeto) en IBM i" en la página 1186 para obtener los detalles.

El usuario debe tener autorización para abrir la cola para salida. La cola **no** debe ser una cola modelo.

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje que se envía y recibe información de retorno una vez completada la solicitud de colocación. Consulte "MQMD (Descriptor de mensaje) en IBM i" en la página 1133 para obtener los detalles.

Si la aplicación proporciona un MQMD version-1 , los datos del mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *MDFMT* en MQMD debe establecerse en FMMDE para indicar que hay un MQMDE presente. Consulte "MQMDE (extensión de descriptor de mensaje) en IBM i" en la página 1179 para obtener más detalles.

PMO (MQPMO)-entrada/salida

Opciones que controlan la acción de MQPUT1.

Consulte "MQPMO (opciones de transferencia de mensajes) en IBM i" en la página 1200 para obtener los detalles.

BUFLEN (entero con signo de 10 dígitos)-entrada

Longitud del mensaje en *BUFFER*.

Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior depende de varios factores; consulte la descripción del parámetro **BUFLEN** de la llamada MQPUT para obtener más detalles.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada

Los datos del mensaje.

Es un almacenamiento intermedio que contiene los datos de mensaje de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *BUFFER* contiene datos de tipo carácter, datos numéricos o ambos, los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** deben establecerse en los valores adecuados para los datos; esto permitirá al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT1 deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2104
(2104, X'838 ') No se reconoce la opción de informe en el descriptor de mensaje.

RC2136
(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2049
(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

RC2241
(2241, X'8C1') El grupo de mensajes no está completo.

RC2242
(2242, X'8C2') El mensaje lógico no está completo.

Si *CMPCOD* es CCFAIL:

RC2001
(2001, X'7D1') La cola base de alias no es un tipo válido.

RC2004
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2013
(2013, X'7DD') Tiempo de caducidad no válido.

RC2014
(2014, X'7DE') Código de comentarios no válido.

RC2017
(2017, X'7E1') No hay más manejadores disponibles.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2024
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2026
(2026, X'7EA') El descriptor de mensaje no es válido.

RC2027
(2027, X'7EB') Falta la cola de respuestas.

RC2029
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

RC2030
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

RC2031
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

RC2035
(2035, X'7F3') No autorizado para el acceso.

RC2042
(2042, X'7FA') El objeto ya está abierto con opciones en conflicto.

RC2043
(2043, X'7FB') Tipo de objeto no válido.

RC2044
(2044, X'7FC') La estructura de descriptor de objeto no es válida.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2047
(2047, X'7FF') Persistencia no válida.

RC2048
(2048, X'800 ') La cola no admite mensajes persistentes.

RC2050
(2050, X'802 ') La prioridad del mensaje no es válida.

RC2051
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2053
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

RC2056
(2056, X'808 ') No hay espacio disponible en disco para la cola.

RC2057
(2057, X'809 ') Tipo de cola no válido.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2061
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

RC2063
(2063, X'80F') Se ha producido un error de seguridad.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2072
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2082
(2082, X'822 ') Cola base alias desconocida.

RC2085
(2085, X'825 ') Nombre de objeto desconocido.

RC2086
(2086, X'826 ') Gestor de colas de objetos desconocido.

RC2087
(2087, X'827 ') Gestor de colas remoto desconocido.

RC2091
(2091, X'82B') Cola de transmisión no local.

RC2092
(2092, X'82C') Cola de transmisión con un uso incorrecto.

RC2097
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

RC2098
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2135
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

RC2136
(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2149
(2149, X'865 ') Estructuras PCF no válidas.

RC2154
(2154, X'86A') Número de registros presentes no válido.

RC2155
(2155, X'86B') Los registros de objeto no son válidos.

RC2156
(2156, X'86C') Los registros de respuesta no son válidos.

RC2158
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

RC2159
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

RC2161
(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2173
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

RC2184
(2184, X'888 ') El nombre de cola remota no es válido.

RC2188
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

RC2189
(2189, X'88D') La resolución de nombres de clúster ha fallado.

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2196
(2196, X'894 ') Cola de transmisión desconocida.

RC2197
(2197, X'895 ') Cola de transmisión predeterminada desconocida.

RC2198
(2198, X'896 ') La cola de transmisión predeterminada no es local.

RC2199
(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

RC2258
(2258, X'8D2') Identificador de grupo no válido.

RC2248
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2249
(2249, X'8C9') Los distintivos de mensaje no son válidos.

RC2250
(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2251
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

RC2252
(2252, X'8CC') Longitud original no válida.

RC2253
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

RC2255
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2257
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2266
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

RC2269
(2269, X'8DD') Error de recurso de clúster.

RC2270
(2270, X'8DE') No hay colas de destino disponibles.

RC2420
(2420) Se ha emitido una llamada MQPUT1 , pero los datos del mensaje contienen una estructura MQEPH que no es válida.

RC2551
(2551, X'9F7') La serie de selección especificada no está disponible.

RC2554
(2554, X'9FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje debe entregarse a un suscriptor con un selector de mensajes ampliado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQPUT1(HCONN : OBJDSC : MSGDSC :
C                                      PMO : BUFLEN : BUFFER :
C                                      CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Establecer atributos de objeto) en IBM i

La llamada MQSET se utiliza para cambiar los atributos de un objeto representado por un descriptor de contexto. El objeto debe ser una cola.

- "Sintaxis" en la página 1382
- "Notas de uso" en la página 1382
- "Parámetros" en la página 1382
- "Declaración RPG" en la página 1386

Sintaxis

MQSET *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Notas de uso

1. Utilizando esta llamada, la aplicación puede especificar una matriz de atributos enteros, o una colección de series de atributos de caracteres, o ambos. Si no se producen errores, todos los atributos especificados se establecen simultáneamente. Si se produce un error (por ejemplo, si un selector no es válido o se intenta establecer un atributo en un valor que no es válido), la llamada falla y no se establece ningún atributo.
2. Los valores de los atributos se pueden determinar utilizando la llamada MQINQ; consulte "MQINQ (Consultar sobre atributos de objeto) en IBM i" en la página 1336 para obtener más detalles.

   Nota: No todos los atributos con valores que se pueden consultar utilizando la llamada MQINQ pueden cambiar sus valores utilizando la llamada MQSET. Por ejemplo, no se pueden establecer atributos de objeto de proceso o de gestor de colas con esta llamada.
3. Los cambios de atributo se conservan entre los reinicios del gestor de colas (aparte de las alteraciones en las colas dinámicas temporales, que no sobreviven a los reinicios del gestor de colas).
4. No puede cambiar los atributos de una cola modelo utilizando la llamada MQSET. Sin embargo, si abre una cola modelo utilizando la llamada MQOPEN con la opción MQOO_SET, puede utilizar la llamada MQSET para establecer los atributos de la cola local dinámica creada por la llamada MQOPEN.
5. Si el objeto que se está definiendo es una cola de clúster, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria.

Para obtener más información sobre los atributos de objeto, consulte:

- "Atributos para colas" en la página 1402
- "Atributos de las listas de nombres" en la página 1431
- "Atributos para definiciones de proceso en IBM i" en la página 1432
- "Atributos del gestor de colas en IBM i" en la página 1434

Parámetros

La llamada MQSET tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de HCONN ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este descriptor de contexto representa el objeto de cola con atributos que se van a establecer. El descriptor de contexto ha sido devuelto por una llamada MQOPEN anterior que especificaba la opción OOSET.

SELCNT (entero con signo de 10 dígitos)-entrada

Recuento de selectores.

Este es el recuento de selectores que se proporcionan en la matriz SELS . Es el número de atributos que se van a establecer. Cero es un valor válido. El número máximo permitido es 256.

SELS (entero con signo de 10 dígitos x SELCNT)-entrada

Matriz de selectores de atributos.

Esta es una matriz de selectores de atributos **SELCNT** ; cada selector identifica un atributo (entero o carácter) con un valor que se va a establecer.

Cada selector debe ser válido para el tipo de cola que representa HOBJ . Sólo se permiten determinados valores IA* y CA*; estos valores se listan más adelante en esta sección.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectores IA*) deben especificarse en INTATR en el mismo orden en el que aparecen estos selectores en SELS. Los valores de atributo que corresponden a selectores de atributo de carácter (selectores CA*) deben especificarse en CHRATR en el mismo orden en el que se producen dichos selectores. Los selectores IA* se pueden intercalar con los selectores CA*; sólo es importante el orden relativo dentro de cada tipo.

No es un error especificar el mismo selector más de una vez; si se realiza esto, el último valor especificado para un selector determinado es el que entra en vigor.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores IA* residen dentro del rango IAFRST a IALAST, y los selectores CA* dentro del rango CAFRST a CALAST.

   Para cada rango, las constantes IALSTU y CALSTU definen el valor más alto que aceptará el gestor de colas.
2. Si todos los selectores IA* aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices SELS y INTATR .

Los atributos que se pueden establecer se listan en la tabla siguiente. No se pueden establecer otros atributos utilizando esta llamada. Para los selectores de atributos CA*, la constante que define la longitud en bytes de la serie necesaria en CHRATR se proporciona entre paréntesis.

Tabla 751. Selectores de atributos MQSET para colas

Selector	Descripción	Nota
CATRGD	Datos de desencadenante (LNTRGD).	“2” en la página 1384
IADIST	Soporte de lista de distribución.	“1” en la página 1384

Tabla 751. Selectores de atributos MQSET para colas (continuación)		
Selector	**Descripción**	**Nota**
IAIGET	Si se permiten operaciones get.	
IIPUT	Si se permiten las operaciones de colocación.	
IATRGC	Control de desencadenante.	“2” en la página 1384
IATRGD	Profundidad del desencadenante.	“2” en la página 1384
IATRGP	Umbral de prioridad del mensaje para activaciones.	“2” en la página 1384
IATRGT	Tipo de desencadenante.	“2” en la página 1384

Notas:

1. Soportado sólo en las plataformas siguientes:
   - AIX
   - IBM i
   - Solaris
   - Windows

   y para clientes de IBM MQ conectados a estos sistemas.
2. No soportado en VSE/ESA.

IACNT (entero con signo de 10 dígitos)-entrada

Recuento de atributos enteros.

Es el número de elementos de la matriz `INTATR` y debe ser como mínimo el número de selectores IA* en el parámetro **SELS** . Cero es un valor válido si no hay ninguno.

INTATR (integ con signo de 10 dígitos x rxIACNT)-entrada

Matriz de atributos enteros.

Esta es una matriz de valores de atributo de entero de `IACNT` . Estos valores de atributo deben estar en el mismo orden que los selectores IA* de la matriz SELS .

CALEN (entero con signo de 10 dígitos)-entrada

Longitud del almacenamiento intermedio de atributos de caracteres.

Esta es la longitud en bytes del parámetro **CHRATR** y debe ser como mínimo la suma de las longitudes de los atributos de caracteres especificados en la matriz SELS . Cero es un valor válido si no hay selectores CA* en SELS.

CHRATR (serie de caracteres de 1 byte x CALEN)-entrada

Atributos de carácter.

Es el almacenamiento intermedio que contiene los valores de atributo de carácter, concatenados entre sí. La longitud del almacenamiento intermedio la proporciona el parámetro **CALEN** .

Los atributos de caracteres deben especificarse en el mismo orden que los selectores CA* de la matriz SELS . La longitud de cada atributo de carácter es fija (consulte SELS). Si el valor que se va a establecer para un atributo contiene menos caracteres no en blanco que la longitud definida del atributo, el valor de CHRATR debe rellenarse a la derecha con espacios en blanco para que el valor del atributo coincida con la longitud definida del atributo.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Si CMPCOD es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCFAIL:

RC2219
(2219, X'8AB') La llamada MQI se ha vuelto a especificar antes de que se completara la llamada anterior.

RC2006
(2006, X'7D6') Longitud de atributos de caracteres no válida.

RC2007
(2007, X'7D7') Serie de atributos de carácter no válida.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2019
(2019, X'7E3') El manejador de objeto no es válido.

RC2020
(2020, X'7E4') El valor del atributo de cola con inhibición-get o con inhibición-put no es válido.

RC2021
(2021, X'7E5') Recuento de atributos enteros no válidos.

RC2023
(2023, X'7E7') Matriz de atributos enteros no válida.

RC2040
(2040, X'7F8') Cola no abierta para el conjunto.

RC2041
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101
(2101, X'835') El objeto se ha dañado.

RC2052
(2052, X'804') La cola se ha suprimido.

RC2058
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162
(2162, X'872') El gestor de colas está concluyendo.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2065
(2065, X'811 ') Recuento de selectores no válido.

RC2067
(2067, X'813 ') El selector de atributos no es válido.

RC2066
(2066, X'812 ') Recuento de selectores demasiado grande.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2075
(2075, X'81B') El valor del atributo trigger-control no es válido.

RC2076
(2076, X'81C') El valor del atributo de profundidad de desencadenante no es válido.

RC2077
(2077, X'81D') El valor del atributo trigger-message-priority no es válido.

RC2078
(2078, X'81E') El valor del atributo de tipo desencadenante no es válido.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSET(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

La definición de prototipo para la llamada es:

```
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 (Establecer propiedad de manejador de mensajes) en IBM i

La llamada MQSETMP establece o modifica una propiedad de un manejador de mensajes.

- "Sintaxis" en la página 1387
- "Notas de uso" en la página 1387
- "Parámetros" en la página 1389
- "Declaración RPG" en la página 1391

Sintaxis

MQSETMP *(Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, CompCode, Reason)*

Notas de uso

- Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
  - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
  - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de IBM MQ .

  Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte "MQBEGIN (Iniciar unidad de trabajo) en IBM i" en la página 1283.
- En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
  - En z/OS, utilice las llamadas siguientes:
    - Los programas por lotes (incluidos los programas IMS batch DL/I) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de IBM MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de IBM MQ como a los recursos pertenecientes a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
    - Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT ROLLBACK` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS .
    - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como `ROLB` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).
  - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato `STRCMTCTL` con el parámetro **`CMTSCOPE(*JOB)`** no debe haberse emitido para el trabajo.
- Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC (Desconectar gestor de colas) en IBM i" en la página 1321 para obtener más detalles.

- Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
  - Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags* en MQMD.
  - Indica si el mensaje forma parte de una unidad de trabajo.
  - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

  El gestor de colas mantiene tres conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:
  - La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
  - La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
  - La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).

  Si la aplicación coloca u obtiene los mensajes como parte de una unidad de trabajo y, a continuación, la aplicación decide restituir la unidad de trabajo, la información de grupo y segmento se restaura al valor que tenía anteriormente:
  - La información asociada con la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT satisfactoria para ese manejador de cola en la unidad de trabajo actual.
  - La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

  Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

  La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo.

  El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

  Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en PMOPT (entero con signo de 10 dígitos)y la opción GMLOGO descrita en GMOPT (entero con signo de 10 dígitos).

  Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:
- Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte HCONN (entero con signo de 10 dígitos)-salida para obtener información sobre el ámbito de los manejadores de conexión.
- Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
- Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse de esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero

lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQSETMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG** .

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra estableciendo una propiedad del descriptor de mensaje; de lo contrario, la llamada falla con el código de razón `RC2009` .

HMSG (entero con signo de 20 dígitos)-entrada

Este es el manejador de mensajes que se va a modificar. El valor ha sido devuelto por una llamada MQCRTMH anterior.

SETOPT (MQSMPO)-entrada

Controlar cómo se establecen las propiedades de mensaje.

Esta estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de mensaje. La estructura es un parámetro de entrada en la llamada MQSETMP. Consulte MQSMPO para obtener más información.

PRNAME (MQCHARV)-entrada

Es el nombre de la propiedad que se va a establecer.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PRPDSC (MQPD)-entrada/salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo:

- qué sucede si la propiedad no está soportada
- a qué contexto de mensaje pertenece la propiedad
- en qué mensajes se copia la propiedad a medida que fluye

Consulte MQPD para obtener más información sobre esta estructura.

TYPE (entero con signo de 10 dígitos)-entrada

El tipo de datos de la propiedad que se está definiendo. Puede ser uno de los siguientes:

TYPBOL
Un booleano. *ValueLength* debe ser 4.

TYPBST
Una serie de bytes. *ValueLength* debe ser cero o mayor.

TYPI8
Un entero con signo de 8 bits. *ValueLength* debe ser 1.

TYPI16
Un entero con signo de 16 bits. *ValueLength* debe ser 2.

TYPI32
Un entero con signo de 32 bits. *ValueLength* debe ser 4.

TYPI64
Un entero con signo de 64 bits. *ValueLength* debe ser 8.

TYPF32
Un número de coma flotante de 32 bits. *ValueLength* debe ser 4.

TYPF64
Un número de coma flotante de 64 bits. *ValueLength* debe ser 8.

TIPSTR
Una serie de caracteres. *ValueLength* debe ser cero o mayor, o el valor especial VLNULL.

TYPNUL
La propiedad existe pero tiene un valor nulo. *ValueLength* debe ser cero.

VALLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes del valor de propiedad en el parámetro *Valor* .

Cero sólo es válido para valores nulos o para series o series de bytes. Cero indica que la propiedad existe pero que el valor no contiene caracteres ni bytes.

El valor debe ser mayor o igual que cero o el siguiente valor especial si el parámetro *Tipo* tiene establecido TYPSTR:

VLNULL
El valor está delimitado por el primer nulo encontrado en la serie. El valor nulo no se incluye como parte de la serie. Este valor no es válido si no se ha establecido también TYPSTR.

Nota: El carácter nulo utilizado para terminar una serie si se establece VLNULL es un nulo del juego de caracteres del valor.

VALUE (serie de bits de 1 byte x VALLEN)-entrada

El valor de la propiedad que se va a establecer. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del valor.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si *ValueLength* es cero, no se hace referencia a *Valor* . En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2421
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFAIL:

RC2204
(2204, X'089C') Adaptador no disponible.

RC2130
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157
(2157, X'86D') Los ASID primario e inicial varían.

RC2004
(2004, X'07D4') El parámetro de valor no es válido.

RC2005
(2005, X'07D5') El parámetro de longitud de valor no es válido.

RC2219
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

RC2460
(2460, X'099C') El puntero de manejador de mensajes no es válido.

RC2499
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

RC2046
(2046, X'07FE') Opciones no válidas o no coherentes.

RC2482
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

RC2442
(2442, X'098A') Nombre de propiedad no válido.

RC2473
(2473, X'09A9') Tipo de datos de propiedad no válido.

RC2472
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

RC2463
(2463, X'099F') Establecer estructura de opciones de propiedad de mensaje no válida.

RC2111
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Consulte “Códigos de retorno para IBM i (ILE RPG)” en la página 1462 para obtener más detalles.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSETMP(HCONN : HMSG : SETOPT :
                                        PRNAME : PRPDSC :
                                        TYPE : VALLEN : VALUE :
                                        CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Recuperar información de estado) en IBM i

Utilice la llamada MQSTAT para recuperar información de estado. El tipo de información de estado devuelta viene determinado por el valor STYPE especificado en la llamada.

- "Sintaxis" en la página 1392
- "Notas de uso" en la página 1392
- "Parámetros" en la página 1392
- "Declaración RPG" en la página 1393

Sintaxis

MQSTAT *(HCONN, STYPE, STAT, CMPCOD, REASON)*

Notas de uso

1. Una llamada a MQSTAT que especifica un tipo de STATAPT devuelve información sobre operaciones MQPUT y MQPUT1 asíncronas anteriores. La estructura MQSTAT pasada en la llamada se completa con la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de CCWARN, en su lugar se devuelve un error posterior con un código de terminación de CCFAIL.
2. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT, se devuelve un CMPCOD de CCOK y REASON de RCNONE.
3. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven utilizando tres contadores: STSPSC, STSPWC y STSPFC. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en lugar de una vez por lista de distribución).
4. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.

Parámetros

La llamada MQSTAT tiene los parámetros siguientes:

Hconn (MQHCONN)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

STYPE (entero con signo de 10 dígitos)-entrada

Tipo de información de estado que se está solicitando. El único valor válido es:

STATAPT
Devuelve información sobre operaciones de transferencia asíncronas anteriores.

STS (MQSTS)-entrada/salida

Estructura de información de estado. Consulte "MQSTS (estructura de informes de estado) en IBM i" en la página 1259 para obtener los detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria.

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2374
(2374, X' 946 ') La salida de API ha fallado

RC2183
(2183, X'887') No se ha podido cargar la salida de la API.

RC2219
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2009
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2203
(2203, X'89B') La conexión está concluyendo.

RC2018
(2018, X'7E2') El descriptor de conexión no es válido.

RC2162
(2162, X'872 ') Se está deteniendo el gestor de colas

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2430
(2430, X'97E') Error con el tipo MQSTAT.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2424
(2424, X' 978 ') Error con estructura MQSTS

RC2195
(2195, X'893') Se ha producido un error inesperado.

RC2298
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Para obtener información detallada sobre estos códigos, consulte:

- Mensajes y códigos de razón

Declaración RPG

```
C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C                     CALLP     MQSTAT(HCONN : ETYPE : ERR :
C                                      CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Registrar suscripción) en IBM i

La llamada MQSUB registra la suscripción de aplicaciones a un tema determinado.

- "Sintaxis" en la página 1394
- "Notas de uso" en la página 1394
- "Parámetros" en la página 1396
- "Declaración RPG" en la página 1399

Sintaxis

MQSUB *(HCONN, SUBDSC, HOBJ, HSUB, CMPCOD, REASON)*

Notas de uso

- La suscripción se realiza a un tema, denominado utilizando el nombre abreviado de un objeto de tema predefinido, el nombre completo de la serie de tema o está formado por la concatenación de dos partes, tal como se describe en Combinación de series de tema.
- El gestor de colas efectúa comprobaciones de seguridad cuando se emite una llamada MQSUB para verificar que el identificador de usuario con el que se está ejecutando la aplicación tiene el nivel de autorización adecuado antes de permitir el acceso. El objeto de tema adecuado se localiza mediante un nombre abreviado que se proporciona en la llamada o el objeto de nombre abreviado más cercano en la jerarquía de temas que se encuentra si se proporciona un nombre largo. Se realiza una comprobación de autorización sobre este objeto de tema para asegurarse de que se ha establecido la autorización para suscribirse y sobre la cola de destino para asegurarse de que se ha establecido la autorización para la salida. Si se utiliza la opción SDMAN, esto significa que se realiza una comprobación de autorización en el nombre de cola gestionada asociado a este objeto de tema, y si se proporciona una cola no gestionada, esto significa que se realiza una comprobación de autorización en la cola representada por el parámetro **HOBJ** .
- El *HOBJ* devuelto en la llamada MQSUB cuando se utiliza la opción SOMAN, se puede consultar para averiguar atributos como, por ejemplo, el umbral de restitución y el nombre de reposición en cola de restitución excesiva. También puede consultar el nombre de la cola gestionada, pero no debe intentar abrir directamente esta cola.
- Las suscripciones se pueden agrupar, lo que permite que se entregue una única publicación al grupo de suscripciones, incluso si más de una suscripción del grupo coincidía con la publicación. Las suscripciones se agrupan utilizando la opción SOGRP y para agrupar suscripciones deben:
  - utilizar la misma cola con nombre (que no utiliza la opción SOMAN) en el mismo gestor de colas-representado por el parámetro **HOBJ** en la llamada MQSUB
  - compartir el mismo *SDCID*
  - ser del mismo *SDSL*

  Estos atributos definen el conjunto de suscripciones que se tienen en cuenta para formar parte de un grupo y que también son los atributos que no se pueden alterar si se agrupa una suscripción. La

modificación de los resultados de *SDSL* en RC2512, y la modificación de cualquiera de los demás (que se puede cambiar si una suscripción no está agrupada) da como resultado RC2515.

- Los campos en MQSD se completan al volver de una llamada MQSUB que utiliza la opción SORES. El MQSD devuelto se puede pasar directamente a una llamada MQSUB que utiliza la opción SOALT con los cambios que necesite realizar en la suscripción aplicada al MQSD. Algunos campos tienen algunas consideraciones especiales, tal como se indica en la tabla.

Tabla 752. Salida MQSD de MQSUB

Nombre de campo en MQSD	**Consideraciones especiales**
Opciones de acceso o creación	Ninguna de estas opciones se establece al volver de la llamada MQSUB. Si más adelante reutiliza el MQSD en una llamada MQSUB, la opción que necesita debe establecerse explícitamente.
Opciones de durabilidad, Opciones de destino, Opciones de registro & Opciones de comodín	Estas opciones se establecerán según corresponda
Opciones de publicación	Estas opciones se establecerán según corresponda, excepto para SONEWP que solo es aplicable a SOCRE.
Otras opciones	Estas opciones no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
ObjectName	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB.
ObjectString	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB. El nombre de tema completo utilizado se devuelve en el campo *SDRO* , si se proporciona un almacenamiento intermedio.
AlternateUserId y AlternateSecurityId	Estos campos de sólo entrada no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
SubExpiry	Al volver de una llamada MQSUB utilizando la opción SORES, este campo se establecerá en la caducidad original de la suscripción y no en el tiempo de caducidad restante. Si, a continuación, reutiliza el MQSD en una llamada MQSUB utilizando la opción SOALT, restablecerá la caducidad de la suscripción para empezar a contar de nuevo.
SubName	Este campo es un campo de entrada en una llamada MQSUB y no se modifica en la salida.
SubUserData y SelectionString	Estos campos de longitud variable se devolverán en la salida de una llamada MQSUB utilizando la opción SORES, si se proporciona un almacenamiento intermedio, y también una longitud de almacenamiento intermedio positiva en *VCHRP*. Si no se proporciona ningún almacenamiento intermedio, sólo se devolverá la longitud en el campo *VCHRL* de MQCHARV.If el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven *VCHRP* bytes en el almacenamiento intermedio proporcionado. Si posteriormente reutiliza el MQSD en una llamada MQSUB utilizando la opción SOALT y no se proporciona un almacenamiento intermedio, pero se proporciona un *VCHRL* distinto de cero, si esa longitud coincide con la longitud existente del campo, no se realizará ninguna modificación en el campo.

Tabla 752. Salida MQSD de MQSUB (continuación)	
Nombre de campo en MQSD	**Consideraciones especiales**
SubCorrelId y PubAccountingToken	Si no utiliza SOSCID, el gestor de colas generará *SDCID* . Si no utiliza SOSETI, el gestor de colas generará *SDACC* . Estos campos se devolverán en MQSD desde una llamada MQSUB utilizando la opción SORES. Si los genera el gestor de colas, el valor generado se devolverá en una llamada MQSUB utilizando la opción SOCRE o SOALT.
PubPriority, SubLevel & PubApplIdentityData	Estos campos se devolverán en MQSD.
ResObjectString	Este campo de sólo salida se devolverá en MQSD si se proporciona un almacenamiento intermedio.

Parámetros

La llamada MQSUB tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

SUBDSC (MQSD)-entrada/salida

Se trata de una estructura que identifica el objeto con uso que está registrando la aplicación. Consulte “MQSD (Descriptor de suscripción) en IBM i” en la página 1241 para obtener más información.

HOBJ (entero con signo de 10 dígitos)-entrada/salida

Este descriptor de contexto representa el acceso que se ha establecido para obtener los mensajes enviados a esta suscripción. Estos mensajes se pueden almacenar en una cola específica o se puede solicitar al gestor de colas que gestione su almacenamiento sin necesidad de una cola específica.

El manejador del objeto.

Si se va a utilizar una cola específica, debe estar asociada a la suscripción en el momento de la creación. Esto se puede hacer de dos maneras:

- Proporcionando este descriptor de contexto al llamar a MQSUB con la opción SDCRT. Si este descriptor de contexto se proporciona como parámetro de entrada en la llamada, debe ser un descriptor de contexto de objeto válido devuelto de una llamada MQOPEN anterior de una cola utilizando al menos una de las opciones OOINP*, OOOOUT (si es una cola remota, por ejemplo) u OOBRW. Si este no es el caso, la llamada falla con RC2019. No puede ser un descriptor de contexto de objeto para una cola alias que se resuelve en un objeto de tema. Si es así, la llamada falla con RC2019
- Utilizando el mandato DEFINE SUB MQSC y proporcionando a ese mandato el nombre de un objeto de cola.

Si el gestor de colas va a gestionar el almacenamiento de los mensajes enviados a esta suscripción, debe indicarlo cuando se crea la suscripción, utilizando la opción SOMAN y estableciendo el valor del parámetro en HONONE. El gestor de colas devuelve el descriptor de contexto como un parámetro de salida en la llamada, y el descriptor de contexto que se devuelve se conoce como descriptor de contexto gestionado. Si se especifica HONONE y no se especifica también SOMAN, la llamada falla con RC2019.

Un descriptor de contexto gestionado devuelto por el gestor de colas se puede utilizar en una llamada MQGET o MQCB, con o sin opciones de examen, en una llamada MQINQ o en MQCLOSE. No se puede utilizar en MQPUT, MQSET o en una MQSUB posterior; el intento de hacerlo falla con RC2039 para MQPUT, RC2040 para MQSET o RC2038 para MQSUB.

Si se utiliza la opción SORES en el campo *OPTS* de la estructura MQSD para reanudar esta suscripción, el descriptor de contexto se puede devolver a la aplicación en este parámetro si se especifica HONONE. Puede utilizar esto tanto si la suscripción está utilizando un descriptor de contexto

gestionado como si no. Puede ser útil para suscripciones creadas utilizando DEFINE SUB si desea el descriptor de contexto para la cola de suscripción definida en el mandato DEFINE SUB. En el caso de que se reanude una suscripción creada administrativamente, la cola se abre con OOINPQ y OOBRW. Si se necesitan otras opciones, la aplicación debe abrir la cola de suscripción de forma explícita y proporcionar el descriptor de objeto en la llamada. Si hay un problema al abrir la cola, la llamada fallará con RC2522. Si se proporciona *HOBJ* , debe ser equivalente a *HOBJ* en la llamada MQSUB original. Esto significa que si se proporciona un descriptor de contexto de objeto devuelto desde una llamada MQOPEN, el descriptor de contexto debe estar en la misma cola que se ha utilizado anteriormente o la llamada falla con RC2019.

Si se está alterando esta suscripción, utilizando la opción SOALT en el campo *OPTS* de la estructura MQSD, se puede proporcionar un *HOBJ* diferente. Las publicaciones que se hayan entregado a la cola identificada anteriormente a través de este parámetro permanecen en dicha cola y es responsabilidad de la aplicación recuperar estos mensajes si el parámetro **HOBJ** representa ahora una cola diferente.

El uso de este parámetro con diversas opciones de suscripción se resume en la tabla siguiente:

Tabla 753. Utilización de hobj con varias opciones de suscripción

Opciones	Hobj	Descripción
SOCRT + SOMAN	Ignorado en la salida	Crea una suscripción con almacenamiento de mensajes gestionado por el gestor de colas.
SOCRT	Descriptor de contexto de objeto válido	Crea una suscripción que facilita una cola específica como destino de los mensajes.
SORES	HONONE	Reanuda una suscripción creada anteriormente (gestionada o no) y hace que el gestor de colas devuelva el descriptor de contexto de objeto para que lo utilice la aplicación.
SORES	Válido, coincidente, descriptor de contexto de objeto	Reanuda una suscripción creada anteriormente que utiliza una cola específica como destino de los mensajes y utiliza un descriptor de contexto de objeto con opciones de apertura específicas.
SOALT + SOMAN	HONONE	Altera una suscripción existente que anteriormente utilizaba una cola específica, para que ahora se gestione.
SOALT	Descriptor de contexto de objeto válido	Altera una suscripción existente para utilizar una cola específica (ya sea de gestionada o de una cola específica diferente).

Tanto si se ha proporcionado como si se ha devuelto, *HOBJ* debe especificarse en las llamadas MQGET posteriores que necesite para recibir las publicaciones.

El descriptor de contexto *HOBJ* deja de ser válido cuando se emite la llamada MQCLOSE en él, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Consulte HCONN para obtener información sobre el ámbito de descriptor de contexto. Un MQCLOSE del descriptor de contexto *HOBJ* no tiene ningún efecto en el descriptor de contexto *HSUB* .

HSUB (entero con signo de 10 dígitos)-salida

Este descriptor de contexto representa la suscripción que se ha realizado. Se puede utilizar para otras dos operaciones:

- Se puede utilizar en una llamada MQSUBRQ posterior para solicitar que se envíen publicaciones cuando se haya utilizado la opción SOPUBR al realizar la suscripción.
- Se puede utilizar en una llamada MQCLOSE subsiguiente para eliminar la suscripción que se ha realizado. El descriptor de contexto *HSUB* deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. El ámbito

del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Un MQCLOSE del descriptor de contexto *HSUB* no tiene ningún efecto en el descriptor de contexto *HOBJ* .

Este descriptor de contexto no se puede pasar a una llamada MQGET o MQCB. Debe utilizar el parámetro **HOBJ** . Si se pasa este descriptor de contexto a cualquier otra llamada IBM MQ , se genera RC2019.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria

CCWARN
Aviso (terminación parcial)

CCFAIL
Llamada fallida

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2019
(2019 X'07E3') El descriptor de contexto de objeto no es válido

RC2046
(2046 X'07FE') Opciones no válidas o no coherentes

RC2085
(2085 X'0825 ') No se puede encontrar el objeto identificado

RC2161
(2161 X'0871 ') Desactivación temporal del gestor de colas

RC2298
(2298 X'08FA') Función no soportada.

RC2424
(2424 X'0978 ') Descriptor de suscripción (MQSD) no válido

RC2425
(2441 X' 979 ') Serie de tema no válida

RC2428
(2428 X'097C') El nombre de suscripción especificado no coincide con las suscripciones existentes

RC2429
(2429 X'097D') El nombre de suscripción existe y lo está utilizando otra aplicación

RC2431
(2431 X'097F') SubUserCampo de datos no válido

RC2432
(2432 X'0980 ') La suscripción existe

RC2434
(2434 X'0982 ') El nombre de suscripción coincide con la suscripción existente

RC2440
(2440 X'0988 ') El campo SubName no es válido

RC2441
(2441 X'0989 ') El campo de serie de objeto no es válido

RC2435
(2435 X'0983 ') El atributo no se puede cambiar utilizando SDALT, o se ha creado la suscripción con SDIMM.

RC2436
(2436 X'0984 ') Opción SODUR no válida

RC2459
(2459, X'99B') Error de sintaxis de serie de selección.

RC2503
(2503 X'09C7') Las llamadas MQSUB están actualmente inhibidas para los temas suscritos.

RC2519
(2519, X'9D7') La serie de selección no es la especificada en la descripción de cómo utilizar una estructura MQCHARV.

RC2551
(2551, X'9F7') La serie de selección especificada no está disponible.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSUB(HCONN : SUBDSC : HOBJ :
C                                     HSUB : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Solicitud de suscripción) en IBM i

La llamada MQSUBRQ realiza una solicitud en una suscripción.

- “Sintaxis” en la página 1399
- “Notas de uso” en la página 1399
- “Parámetros” en la página 1400
- “Declaración RPG” en la página 1401

Sintaxis

MQSUBRQ *(HCONN, HSUB, ACTION, SUBROPT, CMPCOD, REASON)*

Notas de uso

Las siguientes notas de uso se aplican al uso de SRAPUB:

1. Si este verbo se completa correctamente, las publicaciones retenidas que coinciden con la suscripción especificada se han enviado a la suscripción y se pueden recibir utilizando MQGET o MQCB utilizando el HOBJ devuelto en el verbo MQSUB original que ha creado la suscripción.
2. Si el tema suscrito por el verbo MQSUB original que ha creado la suscripción contenía un comodín, es posible que se envíe más de una publicación retenida. El número de publicaciones enviadas como resultado de esta llamada se registra en el campo *SRNMP* de la estructura SBROPT.
3. Si este verbo se completa con un código de razón de RC2437 , no había publicaciones retenidas actualmente para el tema especificado.
4. Si este verbo se completa con un código de razón de RC2525 o RC2526 , actualmente hay publicaciones retenidas para el tema especificado, pero se ha producido un error que significa que no se han podido entregar.
5. La aplicación debe tener una suscripción actual al tema para poder realizar esta llamada. Si la suscripción se ha realizado en una instancia anterior de la aplicación y no está disponible un descriptor de contexto válido para la suscripción, la aplicación debe llamar primero a MQSUB con la opción SORES para obtener un descriptor de contexto para utilizarla en esta llamada.
6. Las publicaciones se envían al destino que está registrado para su uso con la suscripción actual de esta aplicación. Si las publicaciones deben enviarse a otro lugar, la suscripción debe alterarse primero utilizando la llamada MQSUB con la opción SOALT.

Parámetros

La llamada MQSUBRQ tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS , la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *HCONN* :

HCDEFH
Manejador de conexión predeterminado.

HSUB (entero con signo de 10 dígitos)-entrada

Este descriptor de contexto representa la suscripción para la que se va a solicitar una actualización. El valor de *HSUB* se ha devuelto de una llamada MQSUB anterior.

ACTION (entero con signo de 10 dígitos)-entrada

Este parámetro controla la acción concreta que se está solicitando en la suscripción. Se debe especificar uno (y sólo uno) de los siguientes:

SRAPUB

Esta acción solicita que se envíe una publicación de actualización para el tema especificado. Esto se utiliza normalmente si el suscriptor ha especificado la opción SOPUBR en la llamada MQSUB cuando ha realizado la suscripción. Si el gestor de colas tiene una publicación retenida para el tema, se envía al suscriptor. Si no es así, la llamada falla. Si a una aplicación se le envía una publicación que se ha retenido, esto se indica mediante la propiedad de mensaje MQIsRetained de dicha publicación.

Puesto que el tema de la suscripción existente representada por el parámetro **HSUB** puede contener comodines, el suscriptor puede recibir varias publicaciones retenidas.

SBROPT (MQSRO)-entrada/salida

Estas opciones controlan la acción de MQSUBRQ, consulte “MQSRO-Opciones de solicitud de suscripción” en la página 595 para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK
Realización satisfactoria

CCWARN
Aviso (terminación parcial)

CCFAIL
Llamada fallida

Razón (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CPMCOD* es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CPMCOD* es CCFAIL:

RC2298
2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

RC2437
2437 (X'0985 ') No hay publicaciones retenidas almacenadas actualmente para este tema.

RC2046
2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

RC2161
2161 (X'0871 ') Desactivación temporal del gestor de colas

RC2438
2438 (X'0986 ') En la llamada MQSUBRQ, las opciones de solicitud de suscripción MQSRO no son válidas.

Declaración RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSUBRQ(HCONN : HSUB : ACTION :
C                                       SBROPT : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 Atributos de objetos en IBM i

Esta colección de temas lista sólo los objetos IBM MQ que pueden ser objeto de una llamada a función MQINQ, y proporciona detalles de los atributos que se pueden consultar y los selectores que se van a utilizar.

Atributos para colas

Utilice esta información para obtener información sobre los distintos tipos de definiciones de cola y los atributos soportados por cada uno.

Tipos de cola: el gestor de colas da soporte a los siguientes tipos de definición de cola:

Cola local

Se trata de una cola física que almacena mensajes. La cola existe en el gestor de colas local.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminar mensajes de colas de este tipo. El valor del atributo de cola **QType** es QTLOC.

Cola compartida

Se trata de una cola física que almacena mensajes. La cola existe en un repositorio compartido al que pueden acceder todos los gestores de colas que pertenecen al grupo de compartición de colas propietario del repositorio compartido.

Las aplicaciones conectadas a cualquier gestor de colas del grupo de compartición de colas pueden colocar mensajes y eliminar mensajes de colas de este tipo. Estas colas son efectivamente las mismas que las colas locales. El valor del atributo de cola **QType** es QTLOC.

- Las colas compartidas solo están soportadas en z/OS.

Cola de clúster

Se trata de una cola física que almacena mensajes. La cola existe en el gestor de colas local o en uno o varios de los gestores de colas que pertenecen al mismo clúster que el gestor de colas local.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo, independientemente de la ubicación de la cola. Si existe una instancia de la cola en el gestor de colas local, la cola se comporta de la misma forma que una cola local, y las aplicaciones conectadas al gestor de colas local pueden eliminar mensajes de la cola. El valor del atributo de cola **QType** es QTCLUS.

Cola alias

Esto no es una cola física, es un nombre alternativo para una cola local. El nombre de la cola local en el que se resuelve el alias forma parte de la definición de la cola alias.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminarlos de las colas alias: los mensajes se colocan y se eliminan de la cola local en la que se resuelve el alias. El valor del atributo de cola **QType** es QTALS.

Cola remota

No es una cola física, es la definición local de una cola que existe en un gestor de colas remoto. La definición local de la cola remota contiene información que indica al gestor de colas local cómo direccionar los mensajes al gestor de colas remoto.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas remotas: los mensajes se colocan en la cola de transmisión local que se utiliza para direccionar los mensajes al gestor de colas remoto. Las aplicaciones no pueden eliminar mensajes de colas remotas. El valor del atributo de cola **QType** es QTREM.

También se puede utilizar una definición de cola remota para:

- Alias de cola de respuestas

  En este caso, el nombre de la definición es el nombre de una cola de respuestas. Para obtener más información, consulte Definiciones de alias de cola de respuesta.

- Alias de gestor de colas

  En este caso, el nombre de la definición es un alias para un gestor de colas, no el nombre de una cola. Para obtener más información, consulte Definiciones de alias de gestor de colas.

Cola modelo

No es una cola física, es un conjunto de atributos de cola a partir de los cuales se puede crear una cola local.

Los mensajes no se pueden almacenar en colas de este tipo.

Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola. Los tipos de cola a los que se aplica un atributo se indican mediante una "X" en Tabla 754 en la página 1403 y las tablas subsiguientes.

La Tabla 754 en la página 1403 resume los atributos específicos de las colas. Los atributos se describen en orden alfabético.

Los nombres de los atributos que se muestran en la tabla son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más detalles.

En la tabla siguiente, las columnas se aplican de la forma siguiente:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

Tabla 754. Atributos para colas

Atributo	Descripción	Local	Modelo	Alias	Remotos	Clúster
AlterationDate	Fecha en que se modificó por última vez la definición	X		X	X	
AlterationTime	Hora a la que se modificó por última vez la definición	X		X	X	
BackoutRequeueQName	Nombre de cola de reposición en cola de restitución excesivo	X	X			
BackoutThreshold	Umbral de restituciones	X	X			
BaseQName	Nombre de cola al que se resuelve el alias			X		
ClusterChannelNombre	Nombre de canal de clúster emisor	✓	✓			
ClusterName	Nombre del clúster al que pertenece la cola	X		X	X	
ClusterNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola	X		X	X	
CreationDate	Fecha de creación de la cola	X				
CreationTime	Hora en que se creó la cola	X				

Tabla 754. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
CurrentQDepth	Profundidad de cola actual	X				
DefBind	Enlace predeterminado	X		X	X	X
DefinitionType	Tipo de definición de cola	X	X			
DefInputOpenOption	Opción abierta de entrada predeterminada	X	X			
DefPersistence	Persistencia de mensajes predeterminada	X	X	X	X	X
DefPriority	Prioridad de mensajes predeterminada	X	X	X	X	X
DistLists	Soporte de lista de distribución	X	X			
HardenGetBackout	Si se debe mantener un recuento de restituciones preciso	X	X			
InhibitGet	Controla si se permiten operaciones get para la cola	X	X	X		
InhibitPut	Controla si se permiten operaciones de colocación para la cola	X	X	X	X	X
InitiationQName	Nombre de cola de inicio	X	X			
MaxMsgLength	Longitud máxima de mensaje en bytes	X	X			
MaxQDepth	Profundidad máxima de la cola	X	X			
MediaLog	Identidad de la extensión de registro más antigua (o del receptor de diario más antiguo en IBM i) necesario para la recuperación de medios de una cola especificada	✓	✓			
MsgDeliverySequence	Secuencia de entrega de mensajes	X	X			

Tabla 754. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
OpenInputCount	Número de aperturas para entrada	X				
OpenOutputCount	Número de aperturas para salida	X				
ProcessName	Nombre de proceso	X	X			
QDepthHighEvent	Controla si se generan sucesos de profundidad de cola alta	X	X			
QDepthHighLimit	Límite alto para profundidad de cola	X	X			
QDepthLowEvent	Controla si se generan sucesos de profundidad de cola baja	X	X			
QDepthLowLimit	Límite bajo para profundidad de cola	X	X			
QDepthMaxEvent	Controla si se generan sucesos de cola llena	X	X			
QDesc	Descripción de la cola	X	X	X	X	X
QName	Nombre de cola	X		X	X	X
QServiceInterval	Destino para intervalo de servicio de cola	X	X			
SucesoQServiceInterval	Controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto	X	X			
QType	Tipo de cola	X		X	X	X
RemoteQMgrName	Nombre del gestor de colas remoto				X	
RemoteQName	Nombre de cola remota				X	
RetentionInterval	Intervalo de retención	X	X			
Ámbito	Controla si una entrada para la cola también existe en un directorio de célula	X		X	X	
Posibilidad de compartición	Compartibilidad de cola	X	X			
TriggerControl	Activar control	X	X			

Tabla 754. Atributos para colas (continuación)						
Atributo	**Descripción**	**Local**	**Modelo**	**Alias**	**Remotos**	**Clúster**
TriggerData	Datos desencadenantes	X	X			
TriggerDepth	Profundidad de desencadenante	X	X			
TriggerMsgPriority	Prioridad de mensaje de umbral para desencadenantes	X	X			
TriggerType	Tipo de desencadenante	X	X			
Uso	Uso de la cola	X	X			
XmitQName	Nombre de cola de transmisión				X	

IBM i *AlterationDate (serie de caracteres de 12 bytes) en IBM i*

Fecha en la que se cambió por última vez la definición.

Tabla 755. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes (por ejemplo, `1992-09-23¬¬`, donde ¬¬ representa dos caracteres en blanco).

Los valores de determinados atributos (por ejemplo, *`CurrentQDepth`*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *`AlterationDate`*.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

IBM i *AlterationTime (serie de caracteres de 8 bytes) en IBM i*

Hora a la que se modificó por última vez la definición.

Tabla 756. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS` utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, `09.10.20`). La hora es local.

Los valores de determinados atributos (por ejemplo, *`CurrentQDepth`*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *`AlterationTime`*.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

IBM i *BackoutRequeueQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de cola de reposición en cola de restitución excesivo.

Tabla 757. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar dónde deben ir los mensajes que se han restituido. Para todas las demás aplicaciones, aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

Para determinar el valor de este atributo, utilice el selector CABRQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *BackoutThreshold (entero con signo de 10 dígitos) en IBM i*

El umbral de restituciones.

Tabla 758. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar si se debe restituir un mensaje. Para todas las demás aplicaciones, aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

Para determinar el valor de este atributo, utilice el selector IABTHR con la llamada MQINQ.

IBM i *BaseQName (serie de caracteres de 48 bytes) en IBM i*

Es el nombre de la cola a la que resuelve el alias.

Tabla 759. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
		X		

Es el nombre de una cola definida en el gestor de colas local. (Para obtener más información sobre los nombres de cola, consulte la descripción del campo *ODON* en MQOD. La cola es de uno de los tipos siguientes:

QTLOC
Cola local.

QTREM
Definición local de una cola remota.

QTCLUS
Cola de clúster.

Para determinar el valor de este atributo, utilice el selector CABASQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *BaseType (estructura de parámetros enteros) en IBM i*

El tipo de objeto en el que se resuelve el alias.

Tabla 760. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
		X		

Este atributo puede tener uno de los siguientes valores:

OTQ
El tipo de objeto base es una cola

OTTOP
El tipo de objeto base es un tema

IBM i *CFStrucName (serie de caracteres de 12 bytes) en IBM i*

Nombre de estructura de recurso de acoplamiento.

Tabla 761. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el nombre de la estructura del recurso de acoplamiento donde se almacenan los mensajes de la cola. El primer carácter del nombre está en el rango de A a Z, y los caracteres restantes están en el rango de A a Z, de 0 a 9 o en blanco.

El nombre completo de la estructura en el recurso de acoplamiento se obtiene sufiendo el valor del atributo de gestor de colas **QSGName** con el valor del atributo de cola **CFStrucName** .

Este atributo sólo se aplica a las colas compartidas; se ignora si *QSGDisp* no tiene el valor QSGDSH.

Para determinar el valor de este atributo, utilice el selector CACFSN con la llamada MQINQ. La longitud de este atributo la proporciona LNCFSN.

z/OS Este atributo sólo está soportado en z/OS.

ClusterChannelNombre (serie de caracteres de 20 bytes)

`ClusterChannelNombre` es el nombre genérico de los canales de clúster emisor que utilizan esta cola como cola de transmisión. El atributo especifica los canales de clúster emisor han enviado mensajes a un canal de clúster receptor desde esta cola de transmisión de clúster.

Tabla 762. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

La configuración del gestor de colas predeterminado es para todos los canales de clúster emisor para enviar mensajes desde una sola cola de transmisión, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. La configuración predeterminada se puede cambiar modificando el atributo del gestor de colas, **DefClusterXmitQueueType**. El valor predeterminado del atributo es `SCTQ`. Puede cambiar el valor a `CHANNEL`. Si establece el atributo **DefClusterXmitQueueType** en `CHANNEL`, cada canal

de clúster emisor utiliza de forma predeterminada una cola de transmisión de clúster específica, SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

También puede establecer el atributo de cola de transmisión ClusterChannelName en un canal de clúster emisor manualmente. Los mensajes destinados al gestor de colas conectado por el canal de clúster emisor se almacenan en la cola de transmisión que identifica el canal de clúster emisor. No se almacenan en la cola de transmisión de clúster predeterminada. Si establece el atributo ClusterChannelName en blancos, el canal conmuta a la cola de transmisión de clúster predeterminada cuando se reinicia el canal. La cola predeterminada es SYSTEM.CLUSTER.TRANSMIT.*ChannelName* o SYSTEM.CLUSTER.TRANSMIT.QUEUE, en función del valor del atributo DefClusterXmitQueueType del gestor de colas.

Al especificar asteriscos, "*", en **ClusterChannelName**, puede asociar una cola de transmisión con un conjunto de canales de clúster emisor. Los asteriscos pueden estar al principio, al final o en cualquier posición intermedia de la serie de nombre de canal. **ClusterChannelName** está limitado a una longitud de 20 caracteres: MQ_CHANNEL_NAME_LENGTH.

IBM i *ClusterName (serie de caracteres de 48 bytes) en IBM i*

Nombre del clúster al que pertenece la cola.

Tabla 763. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Es el nombre del clúster al que pertenece la cola. Si la cola pertenece a más de un clúster, *ClusterNamelist* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *ClusterName* está en blanco. Al menos uno de *ClusterName* y *ClusterNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CACLN con la llamada MQINQ. La longitud de este atributo la proporciona LNCLUN.

IBM i *ClusterNamelist (serie de caracteres de 48 bytes) en IBM i*

Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola.

Tabla 764. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres a los que pertenece esta cola. Si la cola sólo pertenece a un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *ClusterName* para especificar el nombre del clúster, en cuyo caso *ClusterNamelist* está en blanco. Al menos uno de *ClusterName* y *ClusterNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CACLNL con la llamada MQINQ. La longitud de este atributo la proporciona LNNLN.

IBM i *CreationDate (serie de caracteres de 12 bytes) en IBM i*

Fecha de creación de la cola.

Tabla 765. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

Es la fecha en la que se creó la cola. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes (por ejemplo, `1992-09-23¬¬`, donde ¬¬ representa dos caracteres en blanco).

- En IBM i, la fecha de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector CACRTD con la llamada MQINQ. La longitud de este atributo la proporciona LNCRTD.

IBM i *CreationTime (serie de caracteres de 8 bytes) en IBM i*

Hora a la que se ha creado la cola.

Tabla 766. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

Es la hora en que se ha creado la cola. El formato de la hora es `HH.MM.SS` utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, `09.10.20`). La hora es local.

- En IBM i, la hora de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector CACRTT con la llamada MQINQ. La longitud de este atributo la proporciona LNCRTT.

IBM i *CurrentQDepth (entero con signo de 10 dígitos) en IBM i*

Profundidad de cola actual.

Tabla 767. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

Es el número de mensajes que hay en la cola actualmente. Se incrementa durante una llamada MQPUT y durante la restitución de una llamada MQGET. Se reduce durante una llamada MQGET no examinada y durante la restitución de una llamada MQPUT. El efecto de esto es que el recuento incluye los mensajes que se han colocado en la cola dentro de una unidad de trabajo, pero que todavía no se han confirmado, aunque no sean elegibles para ser recuperados por la llamada MQGET. De forma similar, excluye los mensajes que se han recuperado dentro de una unidad de trabajo utilizando la llamada MQGET, pero que todavía no se han confirmado.

El recuento también incluye los mensajes que han pasado su hora de caducidad pero que todavía no se han descartado, aunque estos mensajes no son elegibles para ser recuperados. Consulte el campo *MDEXP* descrito en “MQMD (Descriptor de mensaje) en IBM i” en la página 1133.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que *CurrentQDepth* supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes- *todos* los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IACDEP con la llamada MQINQ.

IBM i *DefBind (entero con signo de 10 dígitos) en IBM i*

Enlace predeterminado.

Tabla 768. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	X

Este atributo es el enlace predeterminado que se utiliza cuando se especifica OOBNDQ en la llamada MQOPEN y la cola es una cola de clúster. DefBind puede tener uno de los valores siguientes:

BNDOPN
Enlace arreglado por la llamada MQOPEN.

BNDNO
Enlace no arreglado.

BNDGRP
El enlace no se arregla mediante la llamada MQOPEN, sino que se arregla en MQPUT para todos los mensajes de un grupo lógico.

Para determinar el valor de este atributo, utilice el selector IADBND con la llamada MQINQ.

IBM i *DefinitionType (entero con signo de 10 dígitos) en IBM i*

El tipo de definición de la cola.

Tabla 769. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Indica cómo se ha definido la cola. El valor puede ser uno de los siguientes:

QDPRE
Cola permanente predefinida.

La cola es una cola permanente creada por el administrador del sistema; sólo el administrador del sistema puede suprimirla.

Las colas predefinidas se crean utilizando el mandato MQSC de DEFINE y solo se pueden suprimir utilizando el mandato MQSC de DELETE . Las colas predefinidas no se pueden crear a partir de colas modelo.

Los mandatos pueden ser emitidos por un operador o por un usuario autorizado que envía un mensaje de mandato a la cola de entrada de mandatos (consulte el atributo **CommandInputQName** descrito en "Atributos del gestor de colas en IBM i" en la página 1434).

QDPERM
Cola permanente definida dinámicamente.

La cola es una cola permanente creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDPERM para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte "MQCLOSE (Cerrar objeto) en IBM i" en la página 1298 para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica permanente es QSGDQM.

QDTEMP
Cola temporal definida dinámicamente.

La cola es una cola temporal creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDTEMP para el atributo **DefinitionType** .

Este tipo de cola se suprime automáticamente mediante la llamada MQCLOSE cuando la aplicación que la ha creado la cierra.

El valor del atributo **QSGDisp** para una cola dinámica temporal es QSGDQM.

QDSHAR
Cola compartida definida dinámicamente.

La cola es una cola permanente compartida creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDSHAR para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte "MQCLOSE (Cerrar objeto) en IBM i" en la página 1298 para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica compartida es QSGDSH.

Este atributo en una definición de cola modelo no indica cómo se ha definido la cola modelo, porque las colas modelo siempre están predefinidas. En su lugar, el valor de este atributo en la cola modelo se utiliza para determinar el *DefinitionType* de cada una de las colas dinámicas creadas a partir de la definición de cola modelo utilizando la llamada MQOPEN.

Para determinar el valor de este atributo, utilice el selector IADEFT con la llamada MQINQ.

IBM i *DefInputOpenOption (entero con signo de 10 dígitos) en IBM i*

Opción de apertura de entrada predeterminada.

Tabla 770. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Esta es la forma predeterminada en la que se debe abrir la cola para entrada. Se aplica si se especifica la opción OOINPQ en la llamada MQOPEN cuando se abre la cola. Puede tener uno de los valores siguientes:

OOINPX
Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón RC2042 si la cola está abierta actualmente por esta u otra aplicación para entrada de cualquier tipo (OOINPS u OOINPX).

OOINPS
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con OOINPS, pero falla con el código de razón RC2042 si la cola está abierta actualmente con OOINPX.

Para determinar el valor de este atributo, utilice el selector IADINP con la llamada MQINQ.

IBM i *DefPersistence (entero con signo de 10 dígitos) en IBM i*

Persistencia de mensajes predeterminada.

Tabla 771. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Es la persistencia predeterminada de los mensajes en la cola. Se aplica si se especifica PEQDEF en el descriptor de mensaje cuando se coloca el mensaje.

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la llamada MQPUT o MQPUT1 . Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

Puede tener uno de los valores siguientes:

PEPER
El mensaje es persistente.

Esto significa que el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- colas compartidas

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

PENDIENTE
El mensaje no es persistente.

Esto significa que el mensaje normalmente no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante el reinicio del gestor de colas.

En el caso especial de las colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a las anomalías del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Para determinar el valor de este atributo, utilice el selector IADPER con la llamada MQINQ.

IBM i *DefPriority (entero con signo de 10 dígitos) en IBM i*

Prioridad de mensaje predeterminada.

Tabla 772. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Esta es la prioridad predeterminada para los mensajes de la cola. Esto se aplica si se especifica PRQDEF en el descriptor de mensaje cuando el mensaje se coloca en la cola.

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada para el mensaje se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la operación de colocación. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

La forma en que se coloca un mensaje en una cola depende del valor del atributo **MsgDeliverySequence** de la cola:

- Si el atributo **MsgDeliverySequence** es MSPRIO, la posición lógica en la que se coloca un mensaje en la cola depende del valor del campo *MDPRI* en el descriptor de mensaje.
- Si el atributo **MsgDeliverySequence** es MSFIFO, los mensajes se colocan en la cola como si tuvieran una prioridad igual a *DefPriority* de la cola resuelta, independientemente del valor del campo *MDPRI* en el descriptor de mensaje. Sin embargo, el campo *MDPRI* conserva el valor especificado por la aplicación que ha colocado el mensaje. Consulte el atributo **MsgDeliverySequence** descrito en "Atributos para colas" en la página 1402 para obtener más información.

Las prioridades están en el rango de cero (menor) a *MaxPriority* (mayor); consulte el atributo **MaxPriority** descrito en "Atributos del gestor de colas en IBM i" en la página 1434.

Para determinar el valor de este atributo, utilice el selector IADPRI con la llamada MQINQ.

IBM i *DefReadAhead (entero con signo de 10 dígitos) on IBM i*

Especifica el comportamiento de lectura anticipada predeterminado para los mensajes no persistentes entregados al cliente.

Tabla 773. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

DefReadSe puede establecer en uno de los valores siguientes:

RRAHNO
Los mensajes no persistentes no se envían al cliente antes de que una aplicación los solicite. Puede perderse un mensaje no persistente como máximo, si el cliente finaliza de forma anómala.

RAYA
Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Los mensajes no persistentes se pueden perder si el cliente finaliza de forma anómala o si el cliente no consume todos los mensajes que se envían.

SRAHDIS
Lectura anticipada de mensajes no persistentes en no habilitados para esta cola. Los mensajes no se envían al cliente independientemente de si la aplicación cliente solicita la lectura anticipada.

Para determinar el valor de este atributo, utilice el selector IADRAH con la llamada MQINQ.

IBM i *DefPResp (entero con signo de 10 dígitos) en IBM i*

El atributo de tipo de respuesta de colocación predeterminada (DEFPRESP) define el valor utilizado por las aplicaciones cuando el tipo PutResponsedentro de MQPMO se ha establecido en PMRASQ. Este atributo es válido para todos los tipos de cola.

Tabla 774. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

Puede tener uno de los valores siguientes:

SYNC
La operación de colocación se emite de forma síncrona devolviendo una respuesta.

ASYNC
La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

Para determinar el valor de este atributo, utilice el selector IADPRT con la llamada MQINQ.

IBM i *DistLists (entero con signo de 10 dígitos) en IBM i*

Soporte de lista de distribución.

Tabla 775. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Indica si los mensajes de lista de distribución se pueden colocar en la cola. El atributo lo establece un agente de canal de mensajes (MCA) para informar al gestor de colas local si el gestor de colas del otro extremo del canal da soporte a listas de distribución. Este último gestor de colas (denominado "gestor de colas asociado") es el que recibe a continuación el mensaje, después de que un MCA emisor lo haya eliminado de la cola de transmisión local.

El MCA emisor establece el atributo siempre que establece una conexión con el MCA receptor en el gestor de colas asociado. De este modo, el MCA emisor puede hacer que el gestor de colas local coloque en la cola de transmisión sólo los mensajes que el gestor de colas asociado puede procesar correctamente.

Este atributo se utiliza principalmente con colas de transmisión, pero el proceso descrito se realiza independientemente del uso definido para la cola (consulte el atributo **Usage**).

Puede tener uno de los valores siguientes:

DLSUPP
Listas de distribución soportadas.

Esto indica que los mensajes de lista de distribución pueden almacenarse en la cola y transmitirse al gestor de colas asociado en ese formato. Esto reduce la cantidad de proceso necesaria para enviar el mensaje a varios destinos.

DLNSUP
Listas de distribución no soportadas.

Esto indica que los mensajes de lista de distribución no se pueden almacenar en la cola, porque el gestor de colas asociado no da soporte a las listas de distribución. Si una aplicación coloca un mensaje de lista de distribución y ese mensaje se va a colocar en esta cola, el gestor de colas divide el mensaje de lista de distribución y coloca los mensajes individuales en la cola. Esto aumenta la cantidad de proceso necesaria para enviar el mensaje a varios destinos, pero garantiza que el gestor de colas asociado procesará correctamente los mensajes.

Para determinar el valor de este atributo, utilice el selector IADIST con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *HardenGetRestitución (entero con signo de 10 dígitos) en IBM i*

Indica si se debe mantener un recuento de restituciones preciso.

Tabla 776. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para cada mensaje, se mantiene un recuento del número de veces que una llamada MQGET recupera el mensaje dentro de una unidad de trabajo y dicha unidad de trabajo se restituye posteriormente. Este recuento está disponible en el campo *MDBOC* del descriptor de mensaje después de que se haya completado la llamada MQGET.

El recuento de restituciones de mensajes sobrevive cuando se reinicia el gestor de colas. Sin embargo, para asegurarse de que el recuento es preciso, la información debe "guardarse" (grabarse en disco u otro dispositivo de almacenamiento permanente) cada vez que una llamada MQGET recupera un mensaje dentro de una unidad de trabajo para esta cola. Si esto no se hace, y se produce una anomalía del gestor de colas junto con la restitución de la llamada MQGET, es posible que el recuento no se incremente.

Sin embargo, la información de protección para cada llamada MQGET dentro de una unidad de trabajo impone un coste de rendimiento, y el atributo **HardenGetBackout** debe establecerse en QABH sólo si el recuento debe ser preciso.

- En IBM i, el recuento de restituciones de mensajes siempre está protegido, independientemente del valor de este atributo.

Son posibles los siguientes valores:

QABH
Recuento de restituciones recordado.

El refuerzo se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso.

QABNH
Es posible que no se recuerde el recuento de restituciones.

El refuerzo no se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso. Por lo tanto, el recuento puede ser menor de lo que debería ser.

Para determinar el valor de este atributo, utilice el selector IAHGB con la llamada MQINQ.

IBM i *InhibitGet (entero con signo de 10 dígitos) en IBM i*

Controla si se permiten operaciones get para esta cola.

Tabla 777. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

Si la cola es una cola alias, se deben permitir operaciones get para el alias y la cola base en el momento de la operación get, para que la llamada MQGET sea satisfactoria. El valor puede ser uno de los siguientes:

QAGETI
Las operaciones de obtención están inhibidas.

Las llamadas MQGET fallan con el código de razón RC2016. Esto incluye las llamadas MQGET que especifican GMBRWF o GMBRWN.

Nota: Si una llamada MQGET que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **InhibitGet** después de QAGETI no impide que se confirme la unidad de trabajo.

QAGETA
Las operaciones de obtención están permitidas.

Para determinar el valor de este atributo, utilice el selector IAIGET con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *InhibitPut (entero con signo de 10 dígitos) en IBM i*

Controla si se permiten operaciones de colocación para esta cola.

Tabla 778. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, se deben permitir operaciones de colocación para *cada* definición en la vía de acceso (incluidas las definiciones de alias de gestor de colas) en el momento de la operación de colocación, para que la llamada MQPUT o MQPUT1 sea satisfactoria. Puede tener uno de los valores siguientes:

QAPUTI
Las operaciones de colocación están inhibidas.

Las llamadas MQPUT y MQPUT1 fallan con el código de razón RC2051.

Nota: Si una llamada MQPUT que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **`InhibitPut`** posteriormente a QAPUTI no impide que se confirme la unidad de trabajo.

QAPUTA
Las operaciones de colocación están permitidas.

Para determinar el valor de este atributo, utilice el selector IAIPUT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *InitiationQName (serie de caracteres de 48 bytes) en IBM i*

Es el nombre de la cola de inicio.

Tabla 779. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Es el nombre de una cola definida en el gestor de colas local; la cola debe ser de tipo QTLOC. El gestor de colas envía un mensaje desencadenante a la cola de inicio cuando el inicio de la aplicación es necesario como resultado de un mensaje que llega a la cola a la que pertenece este atributo. La cola de inicio debe ser supervisada por una aplicación de supervisor desencadenante que iniciará la aplicación adecuada después de recibir el mensaje desencadenante.

Para determinar el valor de este atributo, utilice el selector CAINIQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *MaxMsgLongitud (entero con signo de 10 dígitos) en IBM i*

Longitud máxima de mensaje en bytes.

Tabla 780. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Este es un límite superior para la longitud del mensaje *físico* más largo que se puede colocar en la cola. Sin embargo, debido a que el atributo de cola **`MaxMsgLength`** se puede establecer independientemente

del atributo de gestor de colas **MaxMsgLength** , el límite superior real para la longitud del mensaje físico más largo que se puede colocar en la cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength** , pero sólo si la aplicación especifica el distintivo MFSEGA en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente, las restricciones de recursos impuestas por el sistema operativo o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

Un intento de colocar en la cola un mensaje demasiado largo falla con el código de razón:

- RC2030 si el mensaje es demasiado grande para la cola
- RC2031 si el mensaje es demasiado grande para el gestor de colas, pero no demasiado grande para la cola

El límite inferior para el atributo **MaxMsgLength** es cero. El límite superior lo determina el entorno:

- En IBM i, la longitud máxima del mensaje es de 100 MB (104 857 600 bytes).

Para obtener más información, consulte el parámetro **BUFLEN** descrito en "MQPUT (transferir mensaje) en IBM i" en la página 1365.

Para determinar el valor de este atributo, utilice el selector IAMLEN con la llamada MQINQ.

IBM i *MaxQDepth (entero con signo de 10 dígitos) en IBM i*

Es la profundidad máxima de la cola.

Tabla 781. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el límite superior definido para el número de mensajes físicos que pueden existir en la cola en cualquier momento. Un intento de colocar un mensaje en una cola que ya contiene mensajes *MaxQDepth* falla con el código de razón RC2053.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que el número real de mensajes físicos en la cola supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes- *todos* los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo es cero o mayor. El límite superior lo determina el entorno.

Nota: Es posible que el espacio de almacenamiento disponible para la cola se agote incluso si hay menos de *MaxQDepth* mensajes en la cola.

Para determinar el valor de este atributo, utilice el selector IAMDEP con la llamada MQINQ.

IBM i *MediaLog (entero con signo de 10 dígitos) en IBM i*

Identidad de la extensión de registro (o receptor de diario en IBM i) necesario para la recuperación de soporte de una cola determinada.

Tabla 782. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

En los gestores de colas donde el registro circular está en uso, el valor se devuelve como una serie nula.

IBM i *MsgDeliverySecuencia (entero con signo de 10 dígitos) en IBM i*

Secuencia de entrega de mensajes.

Tabla 783. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Esto determina el orden en el que la llamada MQGET devuelve los mensajes a la aplicación:

MSFIFO
Los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Esto significa que una llamada MQGET devolverá el *primer* mensaje que cumpla los criterios de selección especificados en la llamada, independientemente de la prioridad del mensaje.

MSPRIO
Los mensajes se devuelven en orden de prioridad.

Esto significa que una llamada MQGET devolverá el mensaje de *prioridad más alta* que satisface los criterios de selección especificados en la llamada. Dentro de cada nivel de prioridad, los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Si los atributos relevantes se cambian mientras hay mensajes en la cola, la secuencia de entrega es la siguiente:

- El orden en el que la llamada MQGET devuelve los mensajes viene determinado por los valores de los atributos **MsgDeliverySequence** y **DefPriority** en vigor para la cola en el momento en que el mensaje llega a la cola:
  - Si *MsgDeliverySequence* es MSFIFO cuando llega el mensaje, el mensaje se coloca en la cola como si su prioridad fuera *DefPriority*. Esto no afecta al valor del campo *MDPRI* en el descriptor de mensaje del mensaje; dicho campo conserva el valor que tenía cuando se colocó el mensaje por primera vez.
  - Si *MsgDeliverySequence* es MSPRIO cuando llega el mensaje, el mensaje se coloca en la cola en el lugar adecuado a la prioridad proporcionada por el campo *MDPRI* en el descriptor de mensaje.

  Si el valor del atributo **MsgDeliverySequence** cambia mientras hay mensajes en la cola, el orden de los mensajes en la cola no cambia.

  Si el valor del atributo **DefPriority** se cambia mientras hay mensajes en la cola, los mensajes no se entregarán necesariamente en orden FIFO, aunque el atributo **MsgDeliverySequence** se establezca en MSFIFO; los que se colocaron en la cola con la prioridad más alta se entregarán primero.

Para determinar el valor de este atributo, utilice el selector IAMDS con la llamada MQINQ.

IBM i *OpenInputRecuento (entero con signo de 10 dígitos) en IBM i*

Es el número de aperturas para entrada.

Tabla 784. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X				

Es el número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola con la llamada MQGET. Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* . Si la cola es una cola compartida, el recuento no incluye las aperturas para la entrada que se han

realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para entrada. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no incluían entrada (por ejemplo, una cola abierta sólo para examinar).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IAOIC con la llamada MQINQ.

IBM i *OpenOutputRecuento (entero con signo de 10 dígitos) en IBM i*

Es el número de aperturas para salida.

Tabla 785. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X				

Es el número de descriptores de contexto que son válidos actualmente para añadir mensajes a la cola con la llamada MQPUT. Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* ; no incluye las aperturas para salida que se han realizado para esta cola en los gestores de colas remotos. Si la cola es una cola compartida, el recuento no incluye las aperturas para la salida que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto para salida una cola alias que se resuelve en esta cola. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no incluían salida (por ejemplo, una cola abierta sólo para consulta).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IAOOC con la llamada MQINQ.

IBM i *ProcessName (serie de caracteres de 48 bytes) en IBM i*

Nombre del proceso.

Tabla 786. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el nombre de un objeto de proceso definido en el gestor de colas local. El objeto de proceso identifica un programa que puede dar servicio a la cola.

Para determinar el valor de este atributo, utilice el selector CAPRON con la llamada MQINQ. La longitud de este atributo la proporciona LNPRON.

IBM i *QDepthHighSuceso (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de profundidad de cola alta.

Tabla 787. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso de profundidad de cola alta indica que una aplicación ha colocado un mensaje en una cola, lo que ha hecho que el número de mensajes de la cola sea mayor o igual que el umbral de profundidad de cola alta (consulte el atributo **QDepthHighLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

El suceso QDepthHighpuede tener uno de dos valores:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAQDHE con la llamada MQINQ.

IBM i *QDepthHighLímite (entero con signo de 10 dígitos) en IBM i*

Límite alto para la profundidad de cola.

Tabla 788. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola alta. Este suceso indica que una aplicación ha colocado un mensaje en una cola, y esto ha hecho que el número de mensajes en la cola sea mayor o igual que el umbral alto de profundidad de cola. Consulte el atributo **QDepthHighEvent** .

El valor se expresa como un porcentaje de la profundidad máxima de cola (atributo**MaxQDepth**) y está en el rango de cero a 100. El valor predeterminado es 80.

Para determinar el valor de este atributo, utilice el selector IAQDHL con la llamada MQINQ.

IBM i *QDepthLowSuceso (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de Profundidad de cola baja.

Tabla 789. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso Profundidad de cola baja indica que una aplicación ha recuperado un mensaje de una cola, lo que ha hecho que el número de mensajes de la cola sea menor o igual que el umbral de profundidad de cola baja (consulte el atributo **QDepthLowLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

El suceso QDepthLowpuede tener uno de los valores siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAQDLE con la llamada MQINQ.

IBM i *QDepthLowLímite (entero con signo de 10 dígitos) en IBM i*

Límite bajo para profundidad de cola.

Tabla 790. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola baja. Este suceso indica que una aplicación ha recuperado un mensaje de una cola y esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral bajo de profundidad de cola. Consulte el atributo **QDepthLowEvent** .

El valor se expresa como un porcentaje de la profundidad máxima de cola (atributo**MaxQDepth**) y está en el rango de cero a 100. El valor predeterminado es de 20.

Para determinar el valor de este atributo, utilice el selector IAQDLL con la llamada MQINQ.

IBM i *QDepthMaxSuceso (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos Cola llena.

Tabla 791. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Un suceso Cola llena indica que una colocación en una cola se ha rechazado porque la cola está llena, es decir, la profundidad de cola ya ha alcanzado su valor máximo.

Nota: El valor de este atributo puede cambiar dinámicamente.

Puede tener uno de los valores siguientes:

EVRDIS
: Informes de sucesos inhabilitados.

EVRENA
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAQDME con la llamada MQINQ.

IBM i *QDesc (serie de caracteres de 64 bytes) en IBM i*

Descripción de cola.

Tabla 792. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X	X	X	X

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo sólo contenga caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAQD con la llamada MQINQ. La longitud de este atributo la proporciona LNQD.

IBM i *QName (serie de caracteres de 48 bytes) en IBM i*

Nombre de cola.

Tabla 793. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Es el nombre de una cola definida en el gestor de colas local. Para obtener más información sobre los nombres de cola, consulte Reglas para la denominación de objetos de IBM MQ. Todas las colas definidas en un gestor de colas comparten el mismo espacio de nombres de cola. Por lo tanto, una cola QTLOC y una cola QTALS no pueden tener el mismo nombre.

Para determinar el valor de este atributo, utilice el selector CAQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *QServiceInterval (entero con signo de 10 dígitos) en IBM i*

Destino del intervalo de servicio de cola.

Tabla 794. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este es el intervalo de servicio utilizado para la comparación para generar sucesos de intervalo de servicio alto y de intervalo de servicio correcto. Consulte el atributo **QServiceIntervalEvent** .

El valor está en unidades de milisegundos y está en el rango de cero a 999.999.999.

Para determinar el valor de este atributo, utilice el selector IAQSI con la llamada MQINQ.

IBM i *QServiceIntervalSuceso (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto.

Tabla 795. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

- Se genera un suceso de intervalo de servicio alto cuando una comprobación indica que no se ha recuperado ningún mensaje de la cola durante al menos el tiempo indicado por el atributo **QServiceInterval** .
- Se genera un suceso de intervalo de servicio correcto cuando una comprobación indica que los mensajes se han recuperado de la cola dentro del tiempo indicado por el atributo **QServiceInterval** .

Nota: El valor de este atributo puede cambiar dinámicamente.

Este atributo puede tener uno de los siguientes valores:

QSIEHI
Sucesos de intervalo de servicio de cola alto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **habilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **inhabilitados**.

QSIEOK
Sucesos de intervalo de servicio de cola correcto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **habilitados**.

QSIENO
No hay sucesos de intervalo de servicio de cola habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto también están **inhabilitados**.

Para las colas compartidas, se ignora el valor de este atributo; se asume el valor QSIENO.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAQSIE con la llamada MQINQ.

IBM i *QSGDisp (entero con signo de 10 dígitos) en IBM i*

Disposición del grupo de compartición de colas.

Tabla 796. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Especifica la disposición de la cola. El valor puede ser uno de los siguientes:

QSGDQM
Disposición del gestor de colas.

El objeto tiene disposición de gestor de colas. Esto significa que la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Es posible que cada gestor de colas del grupo de compartición de colas tenga un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

QSGDCP
Disposición de objeto copiado.

El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero utilizando mandatos MQSC, cada copia se puede modificar para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

QSGDSH
Disposición compartida.

El objeto tiene una disposición compartida. Esto significa que existe en el repositorio compartido una única instancia del objeto que es conocida por todos los gestores de colas del grupo de compartición de colas. Cuando un gestor de colas del grupo accede al objeto, accede a la única instancia compartida del objeto.

Para determinar el valor de este atributo, utilice el selector IAQSGD con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

IBM i *QType (entero con signo de 10 dígitos) en IBM i*

El tipo de cola.

Tabla 797. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	X

Este atributo tiene uno de los siguientes valores:

QTALS
Definición de cola alias.

QTCLUS
Cola de clúster.

QTLOC
Cola local.

QTREM
Definición local de una cola remota.

Para determinar el valor de este atributo, utilice el selector IAQTYP con la llamada MQINQ.

IBM i *RemoteQMgrNombre (serie de caracteres de 48 bytes) en IBM i*

Nombre del gestor de colas remoto.

Tabla 798. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
			X	

Es el nombre del gestor de colas remoto en el que se define la cola *RemoteQName* . Si la cola *RemoteQName* tiene un valor *QSGDisp* de QSGDCP o QSGDSH, *RemoteQMgrName* puede ser el nombre del grupo de compartición de colas propietario de *RemoteQName*.

Si una aplicación abre la definición local de una cola remota, *RemoteQMgrName* no debe estar en blanco y no debe ser el nombre del gestor de colas local. Si *XmitQName* está en blanco, se utiliza la cola local con el mismo nombre que *RemoteQMgrName* como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMgrName*, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName** .

Si esta definición se utiliza para un alias de gestor de colas, *RemoteQMgrName* es el nombre del gestor de colas que se está alias. Puede ser el nombre del gestor de colas local. De lo contrario, si *XmitQName* está en blanco cuando se produce la apertura, debe haber una cola local con el mismo nombre que *RemoteQMgrName* ; esta cola se utiliza como cola de transmisión.

Si esta definición se utiliza para un alias de respuesta, este nombre es el nombre del gestor de colas que debe ser *MDRM*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector CARQMN con la llamada MQINQ. La longitud de este atributo la proporciona LNQMN.

IBM i *RemoteQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de la cola remota.

Tabla 799. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
			X	

Es el nombre de la cola tal como se conoce en el gestor de colas remoto *RemoteQMgrName*.

Si una aplicación abre la definición local de una cola remota, cuando se produce la apertura, *RemoteQName* no debe estar en blanco.

Si esta definición se utiliza para una definición de alias de gestor de colas, cuando se produzca la apertura, *RemoteQName* debe estar en blanco.

Si la definición se utiliza para un alias de respuesta, este nombre es el nombre de la cola que va a ser *MDRQ*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector CARQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *RetentionInterval (entero con signo de 10 dígitos) en IBM i*

Intervalo de retención.

Tabla 800. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Es la hora en la que se debe retener la cola. Una vez transcurrido este tiempo, la cola es apta para su supresión.

El tiempo se mide en horas, contando desde la fecha y hora en que se creó la cola. La fecha de creación de la cola se registra en *CreationDate* y la hora de creación de la cola se registra en el atributo **CreationTime** .

Esta información se proporciona para permitir que una aplicación de mantenimiento o el operador identifique y suprima colas que ya no son necesarias.

Nota: El gestor de colas nunca intenta suprimir colas basadas en este atributo, o impedir la supresión de colas con un intervalo de retención que no ha caducado; es responsabilidad del usuario hacer que se lleve a cabo cualquier acción necesaria.

Se debe utilizar un intervalo de retención realista para evitar la acumulación de colas dinámicas permanentes (consulte *DefinitionType*). Sin embargo, este atributo también se puede utilizar con colas predefinidas.

Para determinar el valor de este atributo, utilice el selector IARINT con la llamada MQINQ.

IBM i *Ámbito (entero con signo de 10 dígitos) en IBM i*

Controla si también existe una entrada para esta cola en un directorio de celdas.

Tabla 801. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X		X	X	

Un servicio de nombres instalable proporciona un directorio de célula. Puede tener uno de los valores siguientes:

SCOQM
Ámbito de gestor de colas.

La definición de cola tiene ámbito de gestor de colas. Esto significa que la definición de la cola no se extiende más allá del gestor de colas que la posee. Para abrir la cola para salida de algún otro gestor de colas, debe especificarse el nombre del gestor de colas propietario o el otro gestor de colas debe tener una definición local de la cola.

SCOCEL
Ámbito de célula.

La definición de cola tiene ámbito de célula. Esto significa que la definición de cola también se coloca en un directorio de célula disponible para todos los gestores de colas de la célula. La cola se puede abrir para salida de cualquiera de los gestores de colas de la célula simplemente especificando el nombre de la cola; no es necesario especificar el nombre del gestor de colas propietario de la cola. Sin embargo, la definición de cola no está disponible para ningún gestor de colas de la célula que también tenga una definición local de una cola con ese nombre, ya que la definición local tiene prioridad.

Un servicio de nombres instalable como LDAP (Lightweight Directory Access Protocol) proporciona un directorio de célula. Tenga en cuenta que IBM MQ ya no da soporte al servicio de nombres DCE (Distributed Computing Environment) que se utilizaba anteriormente para insertar definiciones de cola en un directorio DCE (también ya no está soportado).

El modelo y las colas dinámicas no pueden tener ámbito de célula.

Este valor sólo es válido si se ha configurado un servicio de nombres que da soporte a un directorio de célula.

Para determinar el valor de este atributo, utilice el selector IASCOP con la llamada MQINQ.

El soporte para este atributo está sujeto a las restricciones siguientes:

- En IBM i, el atributo está soportado, pero sólo SCOQM es válido.

IBM i *Compartibilidad (entero con signo de 10 dígitos) en IBM i*

Si la cola se puede compartir para entrada.

Tabla 802. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Esto indica si la cola se puede abrir para entrada varias veces simultáneamente. Puede tener uno de los valores siguientes:

QASHR
La cola es compartible.

Se permiten varias aperturas con la opción OOINPS.

QANSHR
La cola no se puede compartir.

Una llamada MQOPEN con la opción OOINPS se trata como OOINPX.

Para determinar el valor de este atributo, utilice el selector IASHAR con la llamada MQINQ.

IBM i *TriggerControl (entero con signo de 10 dígitos) en IBM i*

Control de desencadenante.

Tabla 803. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Esto controla si los mensajes desencadenantes se graban en una cola de inicio, para hacer que una aplicación se inicie para dar servicio a la cola. Es uno de los siguientes:

TCOFF
: Los mensajes desencadenantes no son necesarios.

  No se deben escribir mensajes desencadenantes para esta cola. El valor de `TriggerType` es irrelevante en este caso.

TCON
: Mensajes desencadenantes necesarios.

  Los mensajes desencadenantes deben escribirse para esta cola, cuando se produzcan los sucesos desencadenantes adecuados.

Para determinar el valor de este atributo, utilice el selector IATRGC con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *TriggerData (serie de caracteres de 64 bytes) en IBM i*

Datos del desencadenante.

Tabla 804. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Son datos de formato libre que el gestor de colas inserta en el mensaje desencadenante cuando un mensaje que llega a esta cola hace que se grabe un mensaje desencadenante en la cola de inicio.

El contenido de estos datos no es significativo para el gestor de colas. Es significativo para la aplicación de supervisor desencadenante que procesa la cola de inicio o para la aplicación iniciada por el supervisor desencadenante.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CATRGD con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET. La longitud de este atributo la proporciona LNTRGD.

IBM i *TriggerDepth (entero con signo de 10 dígitos) en IBM i*

Profundidad del desencadenante.

Tabla 805. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Es el número de mensajes de prioridad *TriggerMsgPriority* o superior que deben estar en la cola antes de que se escriba un mensaje desencadenante. Esto se aplica cuando *TriggerType* se establece en TTDPTH. El valor de *TriggerDepth* es uno o mayor. De lo contrario, este atributo no se utiliza.

Para determinar el valor de este atributo, utilice el selector IATRGD con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *TriggerMsgPrioridad (entero con signo de 10 dígitos) en IBM i*

Prioridad de mensajes de umbral para desencadenantes en IBM MQ for IBM i.

Tabla 806. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Esta es la prioridad de mensaje por debajo de la cual los mensajes no contribuyen a la generación de mensajes desencadenantes (es decir, el gestor de colas ignora estos mensajes al determinar si se debe generar un mensaje desencadenante). *TriggerMsgPriority* puede estar en el rango de cero (más bajo) a *MaxPriority* (más alto; consulte "Atributos del gestor de colas en IBM i" en la página 1434); un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

Para determinar el valor de este atributo, utilice el selector IATRGP con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *TriggerType (entero con signo de 10 dígitos) en IBM i*

Tipo de desencadenante.

Tabla 807. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remotos	Clúster
X	X			

Esto controla las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola. El valor puede ser uno de los siguientes:

TTNONA
No hay mensajes desencadenantes.

No se graban mensajes desencadenantes como resultado de los mensajes de esta cola. Esto tiene el mismo efecto que establecer *TriggerControl* en TCOFF.

TTFRST
Desencadenar mensaje cuando la profundidad de cola va de 0 a 1.

Se graba un mensaje desencadenante siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola cambia de 0 a 1.

TTEVRY
Mensaje desencadenante para cada mensaje.

Se graba un mensaje desencadenante siempre que llega a la cola un mensaje de prioridad *TriggerMsgPriority* o superior.

TTDPTH
Desencadenar mensaje cuando se supere el umbral de profundidad.

Un mensaje desencadenante se graba siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola es igual o superior a *TriggerDepth*. Después de que se

haya escrito el mensaje desencadenante, *TriggerControl* se establece en TCOFF para evitar que se desencadene más hasta que se vuelva a activar explícitamente.

Para determinar el valor de este atributo, utilice el selector IATRGT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i *Uso (entero con signo de 10 dígitos) en IBM i*

Uso de cola.

Tabla 808. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
X	X			

Indica para qué se utiliza la cola. El valor puede ser uno de los siguientes:

USNORM
Uso normal.

Se trata de una cola que las aplicaciones normales utilizan al transferir y obtener mensajes; la cola no es una cola de transmisión.

USTRAN
Cola de transmisión.

Es una cola utilizada para contener mensajes destinados a gestores de colas remotos. Cuando una aplicación normal envía un mensaje a una cola remota, el gestor de colas local almacena el mensaje temporalmente en la cola de transmisión adecuada en un formato especial. A continuación, un agente de canal de mensajes lee el mensaje de la cola de transmisión y lo transporta al gestor de colas remoto. Para obtener más información sobre las colas de transmisión, consulte Colas de transmisión.

Sólo las aplicaciones con privilegios pueden abrir una cola de transmisión para que OOOUT coloque mensajes en ella directamente. Normalmente sólo se espera que las aplicaciones de programa de utilidad lo hagan. Debe tener cuidado de que el formato de datos del mensaje sea correcto (consulte “MQXQH (cabecera de cola de transmisión) en IBM i” en la página 1274); de lo contrario, se pueden producir errores durante el proceso de transmisión. El contexto no se pasa ni se establece a menos que se especifique una de las opciones de contexto PM*.

Para determinar el valor de este atributo, utilice el selector IAUSAG con la llamada MQINQ.

IBM i *XmitQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de la cola de transmisión.

Tabla 809. Tipos de cola a los que se aplica este atributo				
Local	**Modelo**	**Alias**	**Remotos**	**Clúster**
			X	

Si este atributo no está en blanco cuando se produce una apertura, ya sea para una cola remota o para una definición de alias de gestor de colas, especifica el nombre de la cola de transmisión local que se utilizará para reenviar el mensaje.

Si *XmitQName* está en blanco, se utiliza la cola local con el mismo nombre que *RemoteQMgrName* como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMgrName*, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName** .

Este atributo se ignora si la definición se está utilizando como alias de gestor de colas y *RemoteQMgrName* es el nombre del gestor de colas local. Este atributo también se ignora si la definición se utiliza como definición de alias de cola de respuestas.

Para determinar el valor de este atributo, utilice el selector CAXQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

Atributos de las listas de nombres

En este tema se resumen los atributos específicos de las listas de nombres. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos mostrados son los nombres utilizados con las llamadas MQINQ y MQSET.

Descripciones de los atributos

Un objeto de lista de nombres tiene los atributos siguientes:

AlterationDate (serie de caracteres de 12 bytes)

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

AlterationTime (serie de caracteres de 8 bytes)

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

NameCount (entero con signo de 10 dígitos)

Número de nombres en la lista de nombres.

Es mayor o igual que cero. Se define el valor siguiente:

NCMXNL
Número máximo de nombres en una lista de nombres.

Para determinar el valor de este atributo, utilice el selector IANAMC con la llamada MQINQ.

NamelistDesc (serie de caracteres de 64 bytes)

Descripción de lista de nombres.

Se trata de un campo que se puede utilizar para comentarios descriptivos; su valor se establece mediante el proceso de definición. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo sólo contenga caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CALSTD con la llamada MQINQ.

La longitud de este atributo la proporciona LNNLD.

NamelistName (serie de caracteres de 48 bytes)

Nombre de lista de nombres.

Es el nombre de una lista de nombres definida en el gestor de colas local.

Cada lista de nombres tiene un nombre que es diferente de los nombres de otras listas de nombres que pertenecen al gestor de colas, pero puede duplicar los nombres de otros objetos del gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector CALSTN con la llamada MQINQ.

La longitud de este atributo la proporciona LNNLN.

Nombres (serie de caracteres de 48 bytes x NameCount)

Una lista de nombres de *NameCount* .

Cada nombre es el nombre de un objeto definido en el gestor de colas local. Para obtener más información sobre los nombres de objeto, consulte Denominación de objetos IBM MQ.

Para determinar el valor de este atributo, utilice el selector CANAMS con la llamada MQINQ.

La longitud de cada nombre de la lista la proporciona LNOBJN.

IBM i Atributos para definiciones de proceso en IBM i

En este tema se resumen los atributos específicos de las definiciones de proceso. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos mostrados son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más detalles.

Descripciones de los atributos

Un objeto de definición de proceso tiene los atributos siguientes:

AlterationDate (serie de caracteres de 12 bytes)

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

AlterationTime (serie de caracteres de 8 bytes)

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

ApplId (serie de caracteres de 256 bytes)

Identificador de aplicación.

Es una serie de caracteres que identifica la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *ApplId* sea el nombre de un programa ejecutable.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAAPPI con la llamada MQINQ. La longitud de este atributo la proporciona LNPROA.

ApplType (entero con signo de 10 dígitos)

Tipo de aplicación.

Identifica la naturaleza del programa que se va a iniciar en respuesta a la recepción de un mensaje desencadenante. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

ApplType puede tener cualquier valor. Puede utilizar los valores siguientes para los tipos estándar; los tipos de aplicación definidos por el usuario están restringidos a los valores del rango ATUFST a ATULST:

a la(s)CICS
Transacción CICS .

AT400
IBM i .

ATUFST
Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST
Valor más alto para el tipo de aplicación definido por el usuario.

Para determinar el valor de este atributo, utilice el selector IAAPPT con la llamada MQINQ.

EnvData (serie de caracteres de 128 bytes)

Datos de entorno.

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *EnvData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ añade *EnvData* a la lista de parámetros pasada a la aplicación iniciada. La lista de parámetros consta de la estructura MQTMC2 , seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAENVD con la llamada MQINQ. La longitud de este atributo la proporciona LNPROE.

ProcessDesc (serie de caracteres de 64 bytes)

Descripción de proceso.

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAPROD con la llamada MQINQ.

La longitud de este atributo la proporciona LNPROD.

ProcessName (serie de caracteres de 48 bytes)

Nombre del proceso.

Es el nombre de una definición de proceso definida en el gestor de colas local.

Cada definición de proceso tiene un nombre que es diferente de los nombres de otras definiciones de proceso que pertenecen al gestor de colas. Pero el nombre de la definición de proceso puede ser el mismo que los nombres de otros objetos de gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector CAPRON con la llamada MQINQ.

La longitud de este atributo la proporciona LNPRON.

UserData (serie de caracteres de 128 bytes)

Datos de usuario.

Es una serie de caracteres que contiene información de usuario perteneciente a la aplicación que se va a iniciar. Esta información es para que la utilice una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio, o la aplicación que inicia el supervisor desencadenante. La información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *UserData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ pasa *UserData* a la aplicación iniciada como parte de la lista de parámetros. La lista de parámetros consta de la estructura MQTMC2 (que contiene *UserData*), seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAUSRD con la llamada MQINQ. La longitud de este atributo la proporciona LNPROU.

IBM i Atributos del gestor de colas en IBM i

Resumen de los atributos del gestor de colas.

Algunos atributos de gestor de colas son fijos para implementaciones concretas, mientras que otros se pueden cambiar utilizando el mandato MQSC `ALTER QMGR`. Los atributos también se pueden visualizar utilizando el mandato `DISPLAY QMGR`. La mayoría de los atributos del gestor de colas se pueden consultar abriendo un objeto OTQM especial y utilizando la llamada MQINQ con el descriptor de contexto devuelto.

En la tabla siguiente se resumen los atributos que son específicos del gestor de colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos MQSC para obtener más información.

Tabla 810. Atributos para el gestor de colas

Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
AuthorityEvent	Controla si se generan sucesos de autorización (no autorizados)
BridgeEvent	Controla si se generan sucesos de puente IMS

Tabla 810. Atributos para el gestor de colas (continuación)	
Atributo	**Descripción**
ChannelAutoDef	Controla si se permite la definición de canal automática
ChannelAutoDefEvent	Controla si se generan sucesos de definición automática de canal
ChannelAutoDefExit	Nombre de la salida de usuario para la definición automática de canal
ChannelEvent	Controla si se generan sucesos de canal
TipoClusterCache	Controla si la memoria caché de clúster es de tamaño fijo o de tamaño dinámico
ClusterWorkloadData	Datos de usuario para salida de carga de trabajo de clúster
ClusterWorkloadExit	Nombre de salida de usuario para la gestión de carga de trabajo de clúster
ClusterWorkloadLength	Longitud máxima de datos de mensaje pasados a salida de carga de trabajo de clúster
CodedCharSetId	Identificador de juego de caracteres codificado
CommandEvent	Controla si los mensajes de suceso de mandato están en cola
CommandInputQName	Nombre de cola de entrada de mandatos
CommandLevel	Nivel de mandato
ConfigurationEvent	Suceso de configuración
DeadLetterQName	Nombre de la cola de mensajes no entregados
DefClusterXmitQueueTipo	Tipo de cola de transmisión de clúster predeterminado
DefXmitQName	Nombre de cola de transmisión predeterminado
DistLists	Soporte de lista de distribución
InhibitEvent	Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación)
LocalEvent	Controla si se generan sucesos de error locales
LoggerEvent	Controla si se generan sucesos de registro de recuperación
MaxHandles	Número máximo de descriptores de contexto
MaxMsgLength	Longitud máxima de mensaje en bytes
MaxPriority	Prioridad máxima
MaxUncommittedMsgs	Número máximo de mensajes no confirmados dentro de una unidad de trabajo
PerformanceEvent	Controla si se generan sucesos relacionados con el rendimiento
Plataforma	Plataforma en la que se ejecuta el gestor de colas
PubSubMode	Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando
QMgrDesc	Descripción del gestor de colas
QMgrIdentifier	Identificador exclusivo generado internamente del gestor de colas
QMgrName	Nombre del gestor de colas

Tabla 810. Atributos para el gestor de colas (continuación)	
Atributo	**Descripción**
RemoteEvent	Controla si se generan sucesos de error remotos
RepositoryName	Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio
RepositoryNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio
SSLCRLNamelist	Nombre del objeto de lista de nombres que contiene nombres de objetos de información de autenticación (consulte la nota 1)
SSLEvent	Controla si se generan sucesos TLS
SSLKeyRepository	Ubicación del repositorio de claves TLS (consulte la nota 1)
Recuento deSSLKeyReset	Determina el número de bytes no cifrados enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave de cifrado
StartStopEvent	Controla si se generan sucesos de inicio y detención
SyncPoint	Disponibilidad de punto de sincronismo
TraceRouteRecording	Controla el registro de la información de ruta de rastreo para los mensajes
TreeLifeTime	El tiempo de vida, en segundos, de los temas no administrativos
TriggerInterval	Desencadenante-intervalo de mensaje

Notas:

1. Este atributo no se puede consultar utilizando la llamada MQINQ y no se describe en esta sección. Para obtener más información sobre este atributo, consulte Cambiar gestor de colas.

IBM i *AlterationDate (serie de caracteres de 12 bytes) en IBM i*

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es `YYYY-MM-DD`, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

IBM i *AlterationTime (serie de caracteres de 8 bytes) en IBM i*

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es `HH.MM.SS`.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

IBM i *AuthorityEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de autorización (no autorizados).

El atributo AuthorityEvent debe establecerse en uno de los valores siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAAUTE con la llamada MQINQ.

IBM i *BridgeEvent (serie de caracteres) en IBM i*

Este atributo determina si los mensajes de sucesos de puente IMS se colocan en el SYSTEM.ADMIN.CHANNEL.EVENT . Sólo está soportado en z/OS.

IBM i *ChannelAutoDef (entero con signo de 10 dígitos) en IBM i*

Controla si se permite la definición automática de canal.

Este atributo controla la definición automática de canales de tipo CTRCVR y CTSVCN. Tenga en cuenta que la definición automática de canales CTCLSD siempre está habilitada. Puede tener uno de los valores siguientes:

CHADDI
Definición automática de canal inhabilitada.

CHADEN
Definición automática de canal habilitada.

Para determinar el valor de este atributo, utilice el selector IACAD con la llamada MQINQ.

IBM i *ChannelAutoDefEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de definición automática de canal.

Esto se aplica a los canales de tipo CTRCVR, CTSVCN y CTCLSD. Puede tener uno de los valores siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión y rendimiento.

Para determinar el valor de este atributo, utilice el selector IACADE con la llamada MQINQ.

IBM i *ChannelAutoDefExit (serie de caracteres de 20 bytes) en IBM i*

Nombre de la salida de usuario para la definición automática de canal.

Si este nombre no está en blanco y *ChannelAutoDef* tiene el valor CHADEN, se llama a la salida cada vez que el gestor de colas está a punto de crear una definición de canal. Esto se aplica a los canales de tipo CTRCVR, CTSVCN y CTCLSD. A continuación, la salida puede realizar una de las acciones siguientes:

- Permitir la creación de la definición de canal para continuar sin cambios.
- Modifique los atributos de la definición de canal que se crea.
- Suprimir la creación del canal por completo.

Para determinar el valor de este atributo, utilice el selector CACADX con la llamada MQINQ. La longitud de este atributo la proporciona LNEXN.

IBM i *ChannelEvent (serie de caracteres) en IBM i*

Determina si se generan mensajes de sucesos de canal.

Este atributo determina si los mensajes de sucesos de canal se colocan en SYSTEM.ADMIN.CHANNEL.EVENT y, si es así, qué tipo de mensajes se ponen en cola (por ejemplo, 'canal iniciado', 'canal detenido', 'canal no activado'). Antes de la implementación de este atributo, la única forma de impedir que los mensajes de sucesos de canal se colocaran en cola era suprimir la cola de destino.

Este atributo también le permite recopilar sólo sucesos de puente IMS (porque ahora puede desactivar los sucesos de canal, no se colocan en la misma cola). Lo mismo se aplica a los sucesos TLS que también se pueden recopilar sin tener que recopilar también sucesos de canal.

Este atributo también le permite recopilar sólo sucesos significativos (por ejemplo, cuando los canales tienen errores, no cuando se inician y se detienen normalmente).

El valor del atributo ChannelEvent puede ser uno de los siguientes:

- EVREXP (solo se generan los siguientes sucesos de canal: RC2279, RC2283, RC2284, RC2295, RC2296).
- EVRENA (se generan todos los sucesos de canal; es decir, además de los sucesos generados por EVREXP, también se generan los sucesos RC2282y RC2283).
- EVRDIS (no se generan sucesos de canal; este es el valor predeterminado inicial del gestor de colas).

Para determinar el valor de este atributo, utilice el selector IACHNE con la llamada MQINQ.

IBM i *ClusterCacheTipo (serie de caracteres de 32 bytes) en IBM i*

Controla si la memoria caché de clúster es de tamaño fijo o de tamaño dinámico.

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector CACLWD con la llamada MQINQ.

IBM i *ClusterWorkloadDatos (serie de caracteres de 32 bytes) en IBM i*

Datos de usuario para salida de carga de trabajo de clúster.

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector CACLWD con la llamada MQINQ.

IBM i *ClusterWorkloadSalir (serie de caracteres de 20 bytes) en IBM i*

Nombre de salida de usuario para la gestión de carga de trabajo de clúster.

Si este nombre no está en blanco, se llama a la salida cada vez que se transfiere un mensaje a una cola de clúster o se mueve de una cola de clúster emisor a otra. A continuación, la salida puede aceptar la instancia de cola seleccionada por el gestor de colas como destino del mensaje o seleccionar otra instancia de cola.

Para determinar el valor de este atributo, utilice el selector CACLWX con la llamada MQINQ. La longitud de este atributo la proporciona LNEXN.

IBM i *ClusterWorkloadLongitud (entero con signo de 10 dígitos) en IBM i*

Longitud máxima de datos de mensaje pasados a la salida de carga de trabajo de clúster.

Es la longitud máxima de los datos de mensaje que se pasan a la salida de carga de trabajo del clúster. La longitud real de los datos pasados a la salida es el mínimo de lo siguiente:

- Longitud del mensaje.
- El atributo **MaxMsgLength** del gestor de colas.
- El atributo **ClusterWorkloadLength** .

Para determinar el valor de este atributo, utilice el selector IACLWL con la llamada MQINQ.

IBM i *CodedCharSetId (entero con signo de 10 dígitos) en IBM i*

Identificador de juego de caracteres codificados.

Define el juego de caracteres utilizado por el gestor de colas para todos los campos de serie de caracteres definidos en la MQI como, por ejemplo, los nombres de los objetos y la fecha y hora de creación de la

cola. El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de objeto. No se aplica a los datos de aplicación transportados en el mensaje. El valor depende del entorno:

- En IBM i, el valor es el que se establece en el entorno cuando se crea por primera vez el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IACCSI con la llamada MQINQ.

IBM i *CommandEvent (entero) en IBM i*

Controla si los mensajes se colocan en una cola local cuando se emiten mandatos.

Esto controla si los mensajes se graban en una cola de sucesos nueva, SYSTEM.ADMIN.COMMAND.EVENT, siempre que se emitan mandatos. Esta característica es útil para la notificación de seguimiento de mandatos y para el diagnóstico de problemas. Para consultar sobre el atributo de gestor de colas CommandEvent , utilice el nuevo selector de atributos iacev con uno de los valores siguientes:

- EVRENA-los mensajes de suceso de mandato se generan y se colocan en la cola para todos los mandatos satisfactorios.
- Los mensajes de suceso de mandato EVND se generan y se colocan en la cola para todos los mandatos satisfactorios que no sean el mandato DISPLAY (MQSC) y el mandato Inquire (PCF).
- EVRDIS-los mensajes de suceso de mandato no se generan ni se colocan en la cola (este es el valor predeterminado inicial del gestor de colas).

Para determinar el valor de este atributo, utilice el selector CMDEV con la llamada MQINQ.

IBM i *CommandInputQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de cola de entrada de mandatos.

CommandInputQName es el nombre de la cola de entrada de mandatos definida en el gestor de colas local. Es una cola a la que los usuarios pueden enviar mandatos, si están autorizados a hacerlo. El nombre de la cola depende del entorno:

- En IBM i, el nombre de la cola es SYSTEM.ADMIN.COMMAND.QUEUEy solo se le pueden enviar mandatos PCF. Sin embargo, se puede enviar un mandato MQSC a esta cola si el mandato MQSC está entre un mandato PCF de tipo CMESC. Para obtener más información sobre el mandato Escape, consulte Escape.

Para determinar el valor de este atributo, utilice el selector CACMDQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *CommandLevel (entero con signo de 10 dígitos) en IBM i*

Nivel de mandato. Indica el nivel de mandatos de control del sistema soportados por el gestor de colas.

El nivel es uno de los valores siguientes:

CML800
Nivel 800 de mandatos de control del sistema.

Este valor lo devuelven las aplicaciones siguientes:

- IBM MQ for IBM i
  - versión 8.0

CML900
Nivel 900 de mandatos de control del sistema.

Este valor lo devuelven las aplicaciones siguientes:

- IBM MQ for IBM i
  - Versión 9.0

CML910

Nivel 910 de mandatos de control del sistema.

Este valor lo devuelven las aplicaciones siguientes:

- IBM MQ for IBM i
  - versión 9.1

El conjunto de mandatos de control del sistema que corresponde a un valor determinado del atributo **CommandLevel** varía según el valor del atributo **Platform** ; ambos deben utilizarse para decidir qué mandatos de control del sistema están soportados.

Para determinar el valor de este atributo, utilice el selector IACMDL con la llamada MQINQ.

IBM i *ConfigurationEvent en IBM i*

Controla si los sucesos de configuración se generan y se envían a SYSTEM.ADMIN.CONFIG.EVENT EVENT.

El atributo ConfigurationEvent puede ser uno de los valores siguientes:

- EVRENA
- EVRDIS

Si el atributo ConfigurationEvent se establece en EVRENA y runmqsc o PCF emiten correctamente determinados mandatos, los sucesos de configuración se generan y se envían al SYSTEM.ADMIN.CONFIG.EVENT . Se emiten sucesos para los mandatos siguientes, incluso si un mandato alter no cambia el objeto implicado. Los mandatos para los que se generan y envían sucesos de configuración son:

- DEFINE/ALTER AUTHINFO
- DEFINE/ALTER CHANNEL
- DEFINE/ALTER NAMELIST
- DEFINE/ALTER PROCESS
- DEFINE/ALTER QLOCAL (a menos que sea una cola dinámica temporal)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- DELETE AUTHINFO
- DELETE CHANNEL
- DELETE NAMELIST
- DELETE PROCESS
- DELETE QLOCAL (a menos que sea una cola dinámica temporal)
- DELETE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (a menos que el atributo CONFIGEV esté inhabilitado y no se cambie a habilitado)
- REFRESH QMGR
- Una llamada MQSET, que no sea para una cola dinámica temporal.

Los sucesos no se generan (si están habilitados) en las circunstancias siguientes:

- El mandato o la llamada MQSET falla.
- El gestor de colas no puede colocar el mensaje de suceso en la cola de sucesos. El mandato debería completarse correctamente.
- Colas dinámicas temporales.
- Cambios de atributos internos realizados directa o implícitamente (no mediante MQSET o mandato); esto afecta a TRIGGER, CURDEPTH, IPPROCS, OPPROCS, QDPHIEV, QDPLOEV, QDPMAXEV, QSVCIEV.
- Cuando se cambia la cola de sucesos de configuración, aunque se generará un mensaje de suceso para ese cambio cuando se solicite una renovación.

- Agrupación en clúster de cambios mediante los mandatos REFRESH/RESET CLUSTER y RESUME/SUSPEND QMGR.
- Creación o supresión de un gestor de colas.

IBM i *DeadLetterQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de la cola de mensajes no entregados.

Es el nombre de una cola definida en el gestor de colas local. Los mensajes se envían a esta cola si no pueden direccionarse a su destino correcto.

Por ejemplo, los mensajes se colocan en esta cola cuando:

- Llega un mensaje a un gestor de colas, destinado a una cola que todavía no está definida en ese gestor de colas
- Un mensaje llega a un gestor de colas, pero la cola a la que está destinado no puede recibirlo porque, posiblemente:
  - La cola está llena
  - Las solicitudes de colocación están inhibidas
  - El nodo emisor no tiene autorización para colocar mensajes en la cola

Las aplicaciones también pueden colocar mensajes en la cola de mensajes no entregados.

Los mensajes de informe se tratan de la misma forma que los mensajes ordinarios; si el mensaje de informe no se puede entregar a su cola de destino (normalmente la cola especificada por el campo *MDRQ* en el descriptor de mensaje del mensaje original), el mensaje de informe se coloca en la cola de mensajes no entregados (mensaje no entregado).

Nota: Mensajes que han pasado su tiempo de caducidad (consulte el campo *MDEXP* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133) **no** se transfieren a esta cola cuando se descartan. Sin embargo, todavía se genera un mensaje de informe de caducidad (ROEXP) y se envía a la cola *MDRQ* , si lo solicita la aplicación emisora.

Los mensajes no se colocan en la cola de mensajes no entregados cuando la aplicación que ha emitido la solicitud de colocación ha recibido una notificación síncrona del problema con el código de razón devuelto por la llamada MQPUT o MQPUT1 (por ejemplo, un mensaje colocado en una cola local para el que se inhiben las solicitudes de colocación).

Los mensajes de la cola de mensajes no entregados a veces tienen sus datos de mensaje de aplicación con el prefijo de una estructura MQDLH. Esta estructura contiene información adicional que indica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message). Consulte "MQDLH (cabecera Dead-letter) en IBM i" en la página 1086 para obtener más detalles de esta estructura.

Esta cola debe ser una cola local, con un atributo **Usage** de USNORM.

Si un gestor de colas no soporta una cola de mensajes no entregados (undelivered-message) o no se ha definido una, el nombre estará en blanco. Todos los gestores de colas de IBM MQ dan soporte a una cola de mensajes no entregados (undelivered-message), pero de forma predeterminada no está definida.

Si la cola de mensajes no entregados (undelivered-message) no está definida, o está llena, o inutilizable por alguna otra razón, un mensaje que un agente de canal de mensajes le habría transferido se retiene en su lugar en la cola de transmisión.

Para determinar el valor de este atributo, utilice el selector CADLQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

DefClusterXmitQueueTipo (entero con signo de 10 dígitos)

El atributo `DefClusterXmitQueueTipo` controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Los valores de **DefClusterXmitQueueType** son `MQCLXQ_SCTQ` o `MQCLXQ_CHANNEL`.

MQCLXQ_SCTQ

Todos los canales de clúster emisor envían mensajes de SYSTEM.CLUSTER.TRANSMIT.QUEUE. El correlID de los mensajes colocados en la cola de transmisión identifica el canal de clúster emisor al que va destinado el mensaje.

SCTQ se establece cuando se define un gestor de colas. Este comportamiento es implícito en las versiones de IBM WebSphere MQ anteriores a la IBM WebSphere MQ 7.5. En versiones anteriores, el atributo de gestor de colas DefClusterXmitQueueType no estaba presente.

MQCLXQ_CHANNEL

Cada canal de clúster emisor envía mensajes desde una cola de transmisión diferente. Cada cola de transmisión se crea como una cola dinámica permanente de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Si el atributo de gestor de colas, DefClusterXmitQueueTipo, se establece en CHANNEL, la configuración predeterminada cambia a los canales de clúster emisor que se están asociando con colas de transmisión de clúster individuales. Las colas de transmisión son colas dinámicas permanentes creadas a partir de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Cada cola de transmisión está asociada a un canal de clúster emisor. Cuando un canal de clúster emisor presta servicio a una cola de transmisión de clúster, la cola de transmisión contiene mensajes únicamente para un gestor de colas de un clúster. Puede configurar clústeres de modo que cada gestor de colas de un clúster sólo contenga una cola de clúster. En este caso, el tráfico de mensajes de un gestor de colas a cada cola de clúster se transfiere por separado de los mensajes a otras colas.

Para consultar el valor, llame a MQINQo envíe un mandato Consultar gestor de colas (MQCMD_INQUIRE_Q_MGR) PCF, estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Para cambiar el valor, envíe un mandato PCF Cambiar gestor de colas (MQCMD_CHANGE_Q_MGR), estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Referencia relacionada

Cambiar gestor de colas

Consultar gestor de colas

“MQINQ (Consultar sobre atributos de objeto) en IBM i” en la página 1336

La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

IBM i *DefXmitQName (serie de caracteres de 48 bytes) en IBM i*

Nombre de cola de transmisión predeterminado.

Es el nombre de la cola de transmisión que se utiliza para la transmisión de mensajes a gestores de colas remotos, si no hay ninguna otra indicación de qué cola de transmisión utilizar.

Si no hay ninguna cola de transmisión predeterminada, el nombre está totalmente en blanco. El valor inicial de este atributo está en blanco.

Para determinar el valor de este atributo, utilice el selector CADXQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i *DistLists (entero con signo de 10 dígitos) en IBM i*

Soporte de lista de distribución.

Esto indica si el gestor de colas local da soporte a listas de distribución en las llamadas MQPUT y MQPUT1 . Puede tener uno de los valores siguientes:

DLSUPP

Listas de distribución soportadas.

DLNSUP

Listas de distribución no soportadas.

Para determinar el valor de este atributo, utilice el selector IADIST con la llamada MQINQ.

IBM i *InhibitEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación).

Puede tener uno de los valores siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión y rendimiento.

Para determinar el valor de este atributo, utilice el selector IAINHE con la llamada MQINQ.

IBM i *LocalEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de error locales.

El valor puede ser uno de los siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos

Para determinar el valor de este atributo, utilice el selector IALCLE con la llamada MQINQ.

IBM i *LoggerEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de registrador de recuperación.

Puede tener uno de los valores siguientes:

ENABLED
Se generan sucesos de registrador.

DISABLED
Los sucesos de registrador no se generan. Este es el valor predeterminado inicial de los gestores de colas.

Para obtener más información sobre los sucesos, consulte Supervisión y rendimiento.

IBM i *MaxHandles (entero con signo de 10 dígitos) en IBM i*

Número máximo de descriptores de contexto.

Es el número máximo de descriptores de contexto abiertos que cualquier tarea puede utilizar simultáneamente. Cada llamada MQOPEN satisfactoria para una sola cola (o para un objeto que no es una cola) utiliza un descriptor de contexto. Ese descriptor de contexto pasa a estar disponible para su reutilización cuando se cierra el objeto. Sin embargo, cuando se abre una lista de distribución, a cada cola de la lista de distribución se le asigna un descriptor de contexto independiente, y de modo que la llamada MQOPEN utiliza tantos descriptores de contexto como colas hay en la lista de distribución. Esto debe tenerse en cuenta al decidir un valor adecuado para *MaxHandles*.

La llamada MQPUT1 realiza una llamada MQOPEN como parte de su proceso; como resultado, MQPUT1 utiliza tantos manejadores como MQOPEN lo haría, pero los manejadores sólo se utilizan durante la propia llamada MQPUT1 .

El valor está en el rango de 1 a 999.999.999. En IBM i, el valor predeterminado es 256.

Para determinar el valor de este atributo, utilice el selector IAMHND con la llamada MQINQ.

IBM i *MaxMsgLongitud (entero con signo de 10 dígitos) en IBM i*

Longitud máxima de mensaje en bytes.

Es la longitud del mensaje *físico* más largo que puede manejar el gestor de colas. Sin embargo, debido a que el atributo de gestor de colas **MaxMsgLength** se puede establecer independientemente del atributo de cola **MaxMsgLength** , el mensaje físico más largo que se puede colocar en una cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength** , pero sólo si la aplicación especifica el distintivo MFSEGA en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente, las restricciones de recursos impuestas por el sistema operativo o por el entorno en el que se ejecuta la aplicación darán como resultado un límite inferior.

El límite inferior para el atributo **MaxMsgLength** es de 32 KB (32.768 bytes). En IBM i, la longitud máxima del mensaje es de 100 MB (104 857 600 bytes).

Para determinar el valor de este atributo, utilice el selector IAMLEN con la llamada MQINQ.

IBM i *MaxPriority (entero con signo de 10 dígitos) en IBM i*

Prioridad máxima.

Esta es la prioridad de mensaje máxima soportada por el gestor de colas. Las prioridades van de cero (más bajo) a *MaxPriority* (más alto).

Para determinar el valor de este atributo, utilice el selector IAMPRI con la llamada MQINQ.

IBM i *MaxUncommittedMsgs (entero con signo de 10 dígitos) en IBM i*

Número máximo de mensajes no confirmados dentro de una unidad de trabajo.

Es el número máximo de mensajes no confirmados que pueden existir en una unidad de trabajo. El número de mensajes no confirmados es la suma de los siguientes mensajes desde el inicio de la unidad de trabajo actual:

- Mensajes colocados por la aplicación con la opción PMSYP
- Mensajes recuperados por la aplicación con la opción GMSYP
- Mensajes desencadenantes y mensajes de informe COA generados por el gestor de colas para mensajes colocados con la opción PMSYP
- Mensajes de informe COD generados por el gestor de colas para mensajes recuperados con la opción GMSYP

Los mensajes siguientes no se cuentan como no confirmados:

- Mensajes colocados o recuperados por la aplicación fuera de una unidad de trabajo
- Mensajes desencadenantes o mensajes de informe COA/COD generados por el gestor de colas como resultado de mensajes colocados o recuperados fuera de una unidad de trabajo
- Mensajes de informe de caducidad generados por el gestor de colas (aunque la llamada provoque el mensaje de informe de caducidad especificado GMSYP)
- Mensajes de suceso generados por el gestor de colas (aunque la llamada provoque el mensaje de suceso especificado PMSYP o GMSYP)

Nota:

1. Los mensajes de informe de excepción los genera el agente de canal de mensajes (MCA) o la aplicación, por lo que se tratan de la misma forma que los mensajes ordinarios colocados o recuperados por la aplicación.
2. Cuando se coloca un mensaje o segmento con la opción PMSYP, el número de mensajes no confirmados se incrementa en uno independientemente de cuántos mensajes físicos resulten realmente de la colocación. (Puede producirse más de un mensaje físico si el gestor de colas necesita subdividir el mensaje o segmento.)

3. Cuando se coloca una lista de distribución con la opción PMSYP, el número de mensajes no confirmados se incrementa en un *para cada mensaje físico que se genera*. Esto puede ser tan pequeño como uno, o tan grande como el número de destinos en la lista de distribución.

El límite inferior para este atributo es 1; el límite superior es 999 999 999.

Para determinar el valor de este atributo, utilice el selector IAMUNC con la llamada MQINQ.

IBM i *PerformanceEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos relacionados con el rendimiento.

PerformanceEvent puede tener uno de los valores siguientes:

EVRDIS
Informes de sucesos inhabilitados.

EVRENA
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IAPFME con la llamada MQINQ.

IBM i *Plataforma (entero con signo de 10 dígitos) en IBM i*

Plataforma en la que se ejecuta el gestor de colas.

Indica el sistema operativo en el que se ejecuta el gestor de colas. El valor es:

PL400
IBM i.

IBM i *Modalidad PubSub(entero con signo de 10 dígitos) en IBM i*

Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando, permitiendo por lo tanto que las aplicaciones publiquen/suscriban utilizando la interfaz de programación de aplicaciones y las colas que están siendo supervisadas por la interfaz de publicación/suscripción en cola.

Puede tener uno de los valores siguientes:

PSMCP
El motor de publicación/suscripción está ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. La interfaz de publicación/suscripción en cola no se está ejecutando, por lo tanto, no se actúa sobre ningún mensaje que se coloque en las colas supervisadas por la interfaz de publicación/suscripción en cola. Este valor se utiliza para la compatibilidad con WebSphere Message Broker V6 o versiones anteriores que utilizan este gestor de colas, porque debe leer las mismas colas de las que normalmente lee la interfaz de publicación/suscripción en cola.

PSMDS
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola no están ejecutándose. Por lo tanto, no es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. No se actúa sobre los mensajes de publicación/suscripción que se colocan en las colas supervisadas por la interfaz de publicación/suscripción en cola.

PAMÉN
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones y las colas supervisadas por la interfaz de publicación/suscripción en cola. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector PSMODE con la llamada MQINQ.

IBM i *QMgrDesc (serie de caracteres de 64 bytes) en IBM i*

Descripción del gestor de colas.

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

En IBM i, el valor por omisión es blancos.

Para determinar el valor de este atributo, utilice el selector CAQMD con la llamada MQINQ. La longitud de este atributo la proporciona LNQMD.

IBM i *QMgrIdentifier (serie de caracteres de 48 bytes) en IBM i*

Identificador exclusivo generado internamente del gestor de colas.

Es un nombre exclusivo generado internamente para el gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAQMID con la llamada MQINQ. La longitud de este atributo la proporciona LNQMID.

IBM i *QMgrName (serie de caracteres de 48 bytes) en IBM i*

Nombre del gestor de colas.

Es el nombre del gestor de colas local, es decir, el nombre del gestor de colas al que está conectada la aplicación.

Los primeros 12 caracteres del nombre se utilizan para construir un identificador de mensaje exclusivo (consulte el campo *MDMID* descrito en "MQMD (Descriptor de mensaje) en IBM i" en la página 1133). Por lo tanto, los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para que los identificadores de mensajes sean exclusivos en la red de gestores de colas.

Para determinar el valor de este atributo, utilice el selector CAQMN con la llamada MQINQ. La longitud de este atributo la proporciona LNQMN.

IBM i *RemoteEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de error remotos.

El valor puede ser uno de los siguientes:

EVRDIS
: Informes de sucesos inhabilitados.

EVRENA
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IARMTE con la llamada MQINQ.

IBM i *RepositoryName (serie de caracteres de 48 bytes) en IBM i*

Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio.

Es el nombre de un clúster para el que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio para más de un clúster, *RepositoryNamelist* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *RepositoryName* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CARPN con la llamada MQINQ. La longitud de este atributo la proporciona LNQMN.

IBM i *RepositoryNamelist (serie de caracteres de 48 bytes) en IBM i*

Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio.

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio sólo para un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *RepositoryName* para especificar el nombre del clúster, en cuyo caso *RepositoryNamelist* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CARPNL con la llamada MQINQ. La longitud de este atributo la proporciona LNNLN.

IBM i *SSLEvent (serie de caracteres) en IBM i*

Determina si se generan sucesos TLS.

El valor puede ser uno de los siguientes:

- EVRENA (suceso MQINQ/PCF/config) ENABLED (MQSC): se generan sucesos TLS (es decir, se genera el suceso RC2371).
- EVRDIS (suceso MQINQ/PCF/config) DISABLED (MQSC): los sucesos TLS no se generan. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector IASSLE con la llamada MQINQ.

IBM i *SSLKeyResetRecuento (entero) en IBM i*

Determina el número total de bytes no cifrados que se envían y reciben dentro de una conversación TLS, antes de que se renegocie la clave secreta. El número de bytes incluye información de control enviada por el agente de canal de mensajes (MCA).

Este valor sólo lo utilizan los MCA de canal TLS que inician la comunicación desde este gestor de colas (es decir, el MCA de canal emisor en un emparejamiento de canal emisor y receptor).

Si el valor de este atributo es mayor que 0, y las pulsaciones de canal están habilitadas para un canal, la clave secreta también se renegocia antes de que se envíen o reciban los datos después de una pulsación de canal. El recuento de bytes hasta la siguiente renegociación de clave secreta se restablece después de cada renegociación satisfactoria.

El valor puede estar en el rango de 0 a 999.999.999. Un valor de 0 para este atributo indica que la clave secreta nunca se renegocia. Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. Esto es para evitar el coste de proceso de restablecimientos de clave excesivos que se producirían para valores pequeños de restablecimiento de clave secreta TLS.

Cuando el servidor SSL es un gestor de colas de IBM MQ y tanto el restablecimiento de clave secreta como las pulsaciones de canal están habilitadas, la renegociación se produce inmediatamente después de cada pulsación de canal.

Para determinar el valor de este atributo, utilice el selector IASSRC con la llamada MQINQ.

IBM i *StartStopSuceso (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de inicio y detención.

Este atributo puede tener uno de los siguientes valores:

EVRDIS
: Informes de sucesos inhabilitados.

EVRENA
: Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos.

Para determinar el valor de este atributo, utilice el selector IASSE con la llamada MQINQ.

IBM i *SyncPoint (entero con signo de 10 dígitos) en IBM i*

Disponibilidad de punto de sincronismo.

Esto indica si el gestor de colas local soporta unidades de trabajo y sincronización con las llamadas MQGET, MQPUT y MQPUT1 .

SAVL
: Unidades de trabajo y sincronización disponibles.

SPNAVL
: Unidades de trabajo y sincronización no disponibles.

Para determinar el valor de este atributo, utilice el selector IASYNC con la llamada MQINQ.

IBM i *Registro de TraceRoute(entero con signo de 10 dígitos) en IBM i*

Esto controla si la información sobre los mensajes se registra a medida que fluyen a través de un gestor de colas.

El valor puede ser uno de los siguientes:

- RECDD: no se permite ninguna adición para rastrear mensajes de ruta
- RECDQ: los mensajes se colocan en una cola con nombre fijo
- RECDM: determinar utilizando mensaje (este es el valor predeterminado inicial)

Para evitar que el mensaje de ruta de rastreo permanezca en el sistema, establezca un valor de caducidad que sea mayor que cero y especifique la opción de informe RODISC. Para evitar que los mensajes de informe o respuesta permanezcan en el sistema, establezca la opción de informe ROPDAE. Para obtener más información, consulte “Opciones de informe y distintivos de mensaje en IBM i” en la página 1469.

Para determinar el valor de este atributo, utilice el selector IATRGI con la llamada MQINQ.

IBM i *TreeLifeTiempo (entero con signo de 10 dígitos) activado IBM i*

El tiempo de vida, en segundos, de los temas no administrativos.

Los temas no administrativos son los que se crean cuando una aplicación publica en, o se suscribe como, una serie de tema que no existe como nodo administrativo. Cuando este nodo no administrativo ya no tiene ninguna suscripción activa, este parámetro determina cuánto tiempo esperará el gestor de colas antes de eliminar ese nodo. Sólo los temas no administrativos utilizados por una suscripción duradera permanecen después de que se reinicie el gestor de colas.

Especifique un valor en el rango de 0 a 604 000. El valor 0 significa que el gestor de colas no elimina los temas no administrativos. El valor predeterminado inicial del gestor de colas es 1800.

Para determinar el valor de este atributo, utilice el selector IATRLFT con la llamada MQINQ.

IBM i *TriggerInterval (entero con signo de 10 dígitos) en IBM i*

Intervalo de mensaje de desencadenante.

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es TTFRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola, y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenamiento TTFRST incluso si la cola no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte Desencadenamiento de canales.

El valor está en el rango de cero a 999.999.999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector IATRGI con la llamada MQINQ.

Aplicaciones

Esta información describe los programas de ejemplo suministrados con IBM MQ for IBM i para RPG. Además, aprenda a crear aplicaciones ejecutables a partir de los programas que escriba.

creación de una aplicación

Las publicaciones de IBM i describen cómo crear aplicaciones ejecutables a partir de los programas que escribe. En este tema se describen las tareas adicionales y los cambios en las tareas estándar que debe realizar al crear aplicaciones IBM MQ for IBM i para que se ejecuten en IBM i.

Además de codificar las llamadas MQI en el código fuente, debe añadir las sentencias de idioma adecuadas para incluir los archivos de copia IBM MQ for IBM i para el lenguaje RPG. Usted debe familiarizarse con el contenido de estos archivos; sus nombres, y una breve descripción de su contenido se dan en el siguiente texto.

IBM i *IBM MQ copiar archivos en IBM i*

IBM MQ for IBM i proporciona archivos de copia para ayudarle a escribir las aplicaciones en el lenguaje de programación RPG. Son adecuados para su uso con el conjunto de herramientas de desarrollo de WebSphere (5722 WDS) ILE RPG 4 Compiler.

Los archivos de copia que proporciona IBM MQ for IBM i para ayudar con la grabación de salidas de canal se describen en Programas de salida de canal para canales de mensajería.

Los nombres de los archivos de copia de IBM MQ for IBM i para RPG tienen el prefijo CMQ. Tienen un sufijo de G o H. Hay archivos de copia separados que contienen las constantes con nombre y un archivo para cada una de las estructuras. Los archivos de copia se listan en “Consideraciones sobre idiomas” en la página 1029.

Nota: Para ILE RPG/400, se proporcionan como miembros del archivoQRPGLESRC en la biblioteca QMQM.

Las declaraciones de estructura no contienen sentencias DS . Esto permite a la aplicación declarar una estructura de datos (o una estructura de datos de varias apariciones) codificando la sentencia DS y utilizando la sentencia /COPY para copiar en el resto de la declaración:

Para ILE RPG/400 la sentencia es:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure
D MQMD       DS
D/COPY CMQMDG
```

Preparación de los programas para su ejecución

Para crear una aplicación IBM MQ for IBM i ejecutable, debe compilar el código fuente que ha escrito.

Para hacerlo para ILE RPG/400, puede utilizar los mandatos típicos de IBM i , CRTRPGMOD y CRTPGM.

Después de crear *MODULE, debe especificar BNDSRVPGM(QMQM/LIBMQM) en el mandato CRTPGM. Esto incluye los diversos procedimientos de IBM MQ en el programa.

Asegúrese de que la biblioteca que contiene los archivos de copia (QMQM) esté en la lista de bibliotecas cuando realice la compilación.

Para obtener más información sobre las consideraciones de programación, incluidas las modalidades de cliente, consulte “Consideraciones sobre idiomas” en la página 1029.

Interfaces para el gestor de puntos de sincronismo externo de IBM i

IBM MQ for IBM i utiliza el control de compromiso de IBM i nativo como coordinador de punto de sincronismo externo.

Consulte la publicación *IBM i Programming: Backup and Recovery Guide* para obtener más información sobre las prestaciones de control de compromiso de IBM i.

Para iniciar las funciones de control de confirmación de IBM i, utilice el mandato del sistema STRCMTCTL. Para finalizar el control de confirmación, utilice el mandato ENDCMTCTL del sistema.

Nota: El valor predeterminado del *ámbito de definición de confirmación* es *ACTGRP. Esto se debe definir como *JOB en IBM MQ para IBM i. Por ejemplo:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

Si llama a MQPUT, MQPUT1o MQGET, especificando PMSYP o GMSYP, después de iniciar el control de compromiso, IBM MQ for IBM i se añade a sí mismo como recurso de compromiso de API a la definición de compromiso. Normalmente, esto se lleva a cabo en la primera llamada de un trabajo. Mientras haya algún recurso de confirmación de API registrado bajo una definición de confirmación concreta, no podrá finalizar el control de confirmación para dicha definición.

IBM MQ for IBM i elimina su registro como recurso de confirmación de API cuando se desconecta del gestor de colas, siempre que no haya operaciones MQI pendientes en la unidad de trabajo actual.

Si se desconecta del gestor de colas mientras hay operaciones MQPUT, MQPUT1 o MQGET pendientes en la unidad de trabajo, IBM MQ for IBM i continúa registrado como recurso de confirmación de API, para que se le notifique la siguiente confirmación o retrotracción. Cuando se alcanza el siguiente punto de sincronismo, IBM MQ confirma o retrotrae los cambios, según sea necesario. Es posible que una aplicación se desconecte y se vuelva a conectar a un gestor de colas durante una unidad de trabajo activa y realice más operaciones MQGET y MQPUT dentro de la misma unidad de trabajo (esto es una desconexión pendiente).

Si intenta emitir un mandato del sistema ENDCMTCTL para esta definición de compromiso, se emite el mensaje CPF8355, lo que indica que hay cambios pendientes activos. Este mensaje también aparece en las anotaciones de trabajo cuando el trabajo finaliza. Para evitarlo, asegúrese de que confirma o retrotrae todas las operaciones de IBM MQ pendientes y de que se desconecta del gestor de colas. Por lo tanto, el uso de los mandatos COMMIT o ROLLBACK antes de ENDCMTCTL debe permitir que el control de finalización de compromiso se complete satisfactoriamente.

Cuando se utiliza el control de compromiso de IBM i como coordinador de punto de sincronismo externo, es posible que no se emitan las llamadas MQCMIT, MQBACK y MQBEGIN. Las llamadas a estas funciones fallan con el código de razón RC2012.

Para confirmar o retrotraer (esto es, restituir) la unidad de trabajo, utilice uno de los siguientes lenguajes de programación que dan soporte al control de confirmación. Por ejemplo:

- Mandatos CL: COMMIT y ROLLBACK
- Funciones de programación ILE C: _Rcommit y _Rrollback
- RPG/400: COMMIT y ROLBK
- COBOL/400: COMMIT y ROLLBACK

Puntos de sincronización en CICS para las aplicaciones de IBM i

IBM MQ for IBM i participa en unidades de trabajo con CICS. Puede utilizar la MQI dentro de una aplicación CICS para colocar y obtener mensajes dentro de la unidad de trabajo actual.

Puede utilizar el mandato EXEC CICS SYNCPOINT para establecer un punto de sincronización que incluya las operaciones de IBM MQ for IBM i. Para restituir todos los cambios al punto de sincronización anterior, puede utilizar el mandato EXEC CICS SYNCPOINT ROLLBACK.

Si utiliza MQPUT, MQPUT1o MQGET con la opción PMSYP, o GMSYP, establecida en una aplicación CICS , no puede cerrar la sesión CICS hasta que IBM MQ for IBM i haya eliminado su registro como recurso de

compromiso de API. Por lo tanto, debe confirmar o restituir las operaciones de colocación u obtención pendientes antes de desconectarse del gestor de colas. Esto le permitirá cerrar la sesión de CICS.

Programas de ejemplo en IBM i

En este tema se describen los programas de ejemplo suministrados con IBM MQ for IBM i para RPG. Los ejemplos muestran los usos típicos de la interfaz de cola de mensajes (MQI).

Los ejemplos no están pensados para mostrar técnicas de programación generales, por lo que se ha omitido alguna comprobación de errores que puede que desee incluir en un programa de producción. Sin embargo, estos ejemplos son adecuados para su uso como base para sus propios programas de colas de mensajes.

El código fuente de todos los ejemplos se facilita con el producto; dicho fuente incluye comentarios que explican las técnicas de gestión de colas de mensajes mostradas en los programas.

Hay un conjunto de programas de ejemplo ILE:

1. **Programas que utilizan llamadas con prototipo a la MQI (llamadas de enlace estático)**

   El origen existe en QMQMSAMP/QRPGLESRC. Los miembros se denominan AMQ3xxx4, donde xxx indica la función de ejemplo. Los miembros de copia existen en QMQM/QRPGLESRC. Cada nombre de miembro tiene un sufijo de G o H.

Tabla 811 en la página 1451 proporciona una lista completa de los programas de ejemplo que se entregan con IBM MQ for IBM iy muestra los nombres de los programas en cada uno de los lenguajes de programación soportados. Observe que todos sus nombres empiezan con el prefijo AMQ, el cuarto carácter del nombre indica el lenguaje de programación.

Tabla 811. Nombres de los programas de ejemplo

	RPG (ILE)
Ejemplos de colocación	AMQ3PUT4
Examinar ejemplos	AMQ3GBR4
Obtener ejemplos	AMQ3GET4
Ejemplos de solicitud	AMQ3REQ4
Ejemplos de eco	AMQ3ECH4
Consultar ejemplos	AMQ3INQ4
Establecer ejemplos	AMQ3SET4
Ejemplo de supervisor desencadenante	AMQ3TRG4
Ejemplo de servidor desencadenante	AMQ3SRV4

Además de estos, la opción de ejemplo IBM MQ for IBM i incluye un archivo de datos de ejemplo, AMQSDATA, que se puede utilizar como entrada para determinados programas de ejemplo y programas CL de ejemplo que muestran tareas de administración. Los ejemplos de CL se describen en Administración de IBM i . Puede utilizar el programa CL de ejemplo para crear colas que utilizar con los programas de ejemplo descritos en este tema.

Para obtener información sobre cómo ejecutar los programas de ejemplo, consulte “Preparación y ejecución de los programas de ejemplo en IBM i” en la página 1452.

Características demostradas en los programas de ejemplo en IBM i

Una tabla que muestra las técnicas demostradas por los programas de ejemplo de IBM MQ for IBM i .

Algunas técnicas se utilizan en más de un programa de ejemplo, pero en la tabla se indica un solo programa. Todos los ejemplos abren y cierran colas utilizando las llamadas MQOPEN y MQCLOSE, por lo que estas técnicas no se listan por separado en la tabla.

Tabla 812. Programas de ejemplo que demuestran el uso de MQI

Técnica	**RPG (ILE)**
Utilización de las llamadas MQCONN y MQDISC	AMQ3ECH4 o AMQ3INQ4
Conexión y desconexión implícitas	AMQ3PUT4
Colocación de mensajes mediante la llamada MQPUT	AMQ3PUT4
Colocación de un único mensaje utilizando la llamada MQPUT1	AMQ3ECH4 o AMQ3INQ4
Respuesta a un mensaje de solicitud	AMQ3INQ4
Obtención de mensajes (sin espera)	AMQ3GBR4
Obtención de mensajes (espera con límite de tiempo)	AMQ3GET4
Obtención de mensajes (con conversión de datos)	AMQ3ECH4
Examinar una cola	AMQ3GBR4
Utilización de una cola de entrada compartida	AMQ3INQ4
Utilización de una cola de entrada exclusiva	AMQ3REQ4
Utilización de la llamada MQINQ	AMQ3INQ4
Utilización de la llamada MQSET	AMQ3SET4
Utilización de una cola de respuesta	AMQ3REQ4
Solicitud de mensajes de excepción	AMQ3REQ4
Aceptación de un mensaje truncado	AMQ3GBR4
Utilización de un nombre de cola resuelto	AMQ3GBR4
Proceso de desencadenante	AMQ3SRV4 o AMQ3TRG4

Nota: Todos los programas de ejemplo producen un archivo de spool que contiene los resultados del proceso.

Preparación y ejecución de los programas de ejemplo en IBM i

Para poder ejecutar los programas de ejemplo IBM MQ for IBM i , debe compilarlos como lo haría con cualquier otra aplicación IBM MQ for IBM i . Para ello, puede utilizar los mandatos CRTRPGMOD y CRTPGM de IBM i .

Al crear los programas AMQ3xxx4 , debe especificar BNDSRVPGM (QMQM/LIBMQM) en el mandato CRTPGM. Al hacerlo, se incluyen los diversos procedimientos de IBM MQ en el programa.

Los programas de ejemplo se proporcionan en la biblioteca QMQMSAMP como miembros de QRPGLESRC. Utilizan los archivos de copia proporcionados en la biblioteca QMQM, por lo que debe asegurarse de que esta biblioteca esté en la lista de bibliotecas cuando los compile. El compilador RPG proporciona mensajes informativos porque los ejemplos no utilizan muchas de las variables declaradas en los archivos de copia.

Ejecución de los programas de ejemplo

Puede utilizar sus propias colas al ejecutar los ejemplos, o puede compilar y ejecutar AMQSAMP4 para crear algunas colas de ejemplo. El origen de este programa se suministra en el archivo QCLSRC de la biblioteca QMQMSAMP. Se puede compilar utilizando el mandato CRTCLPGM.

Para llamar a uno de los programas de ejemplo, utilice un mandato como:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Donde `Queue_Name` y `Queue_Manager_Name` deben tener 48 caracteres de longitud, lo que se consigue rellenando `Queue_Name` y `Queue_Manager_Name` con el número necesario de espacios en blanco.

Para los programas de ejemplo Inquire y Set, las definiciones de ejemplo creadas por AMQSAMP4 hacen que se desencadenen las versiones C de estos ejemplos. Si desea desencadenar las versiones de RPG, debe cambiar las definiciones de proceso SYSTEM.SAMPLE.ECHOPROCESS y SYSTEM.SAMPLE.INQPROCESS y SYSTEM.SAMPLE.SETPROCESS. Puede utilizar el mandato CHGMQMPRC (descrito en Cambiar proceso de MQ (CHGMQMPRC)) para hacerlo, o edite y ejecute AMQSAMP4 con la definición alternativa.

El programa de ejemplo Put en IBM i

El programa de ejemplo de colocación, AMQ3PUT4, coloca mensajes en una cola utilizando la llamada MQPUT.

Para iniciar el programa, invóquelo pasándole el nombre de la cola de destino como parámetro. El programa coloca un conjunto de mensajes fijos en la cola; estos mensajes se toman del bloque de datos al final del código fuente del programa. Un programa de colocación de ejemplo es AMQ3PUT4 en la biblioteca QMQMSAMP.

Utilizando este programa de ejemplo, el mandato es:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Donde `Queue_Name` y `Queue_Manager_Name` deben tener 48 caracteres de longitud, lo que se consigue rellenando `Queue_Name` y `Queue_Manager_Name` con el número necesario de espacios en blanco.

Diseño del programa de transferencia de ejemplo

El programa utiliza la llamada MQOPEN con la opción OOOUT para abrir la cola de destino para transferir mensajes. Los resultados se envían a un archivo de spool. Si no puede abrir la cola, el programa escribe un mensaje de error que contiene el código de razón devuelto por la llamada MQOPEN. Para que el programa sea sencillo, en esta y en las siguientes llamadas MQI, se utilizarán los valores predeterminados para numerosas opciones.

Para cada línea de datos contenida en el código fuente, el programa lee el texto en un almacenamiento intermedio y utiliza la llamada MQPUT para crear un mensaje de datagrama que contenga el texto de esa línea. El programa continúa hasta llegar al final de la entrada o hasta que falla la llamada MQPUT. Si el programa alcanza el final de la entrada, cierra la cola con la llamada MQCLOSE.

El programa de ejemplo Examinar en IBM i

El programa de ejemplo Examinar, AMQ3GBR4, examina los mensajes de una cola utilizando la llamada MQGET.

El programa recupera copias de todos los mensajes de la cola que especifique al llamar al programa; los mensajes permanecen en la cola. Puede utilizar la cola proporcionada SYSTEM.SAMPLE.LOCAL; ejecute primero el programa de ejemplo Put para colocar algunos mensajes en la cola. Puede utilizar la cola SYSTEM.SAMPLE.ALIAS, que es un nombre de alias para la misma cola local. El programa continúa hasta que llega al final de la cola o hasta que falla la llamada MQI.

Un ejemplo de un mandato para llamar al programa RPG es:

```
CALL PGM(QMQMSAMP/AMQ3GBR4) PARM('Queue_Name','Queue_Manager_Name')
```

Donde `Queue_Name` y `Queue_Manager_Name` deben tener 48 caracteres de longitud, lo que se consigue rellenando `Queue_Name` y `Queue_Manager_Name` con el número necesario de espacios en blanco. Por lo tanto, si utiliza SYSTEM.SAMPLE.LOCAL como cola de destino, necesitará 29 caracteres en blanco.

Diseño del programa de ejemplo de examen

El programa abre la cola de destino utilizando la llamada MQOPEN con la opción OOBRW. Si no puede abrir la cola, el programa escribe un mensaje de error en su archivo de spool, que contiene el código de razón devuelto por la llamada MQOPEN.

Por cada mensaje de la cola, el programa utiliza la llamada MQGET para copiar dicho mensaje de la cola y luego muestra los datos contenidos en el mensaje. La llamada MQGET utiliza estas opciones:

GMBRWN
: Después de la llamada MQOPEN, el cursor para examinar se coloca lógicamente antes del primer mensaje de la cola, por lo que esta opción hace que se devuelva el *primer* mensaje cuando se realiza la llamada por primera vez.

GMNWT
: Si no hay ningún mensaje en la cola, el programa no espera.

GMATM
: La llamada MQGET especifica un búfer de tamaño fijo. Si un mensaje es más largo que este búfer, el programa mostrará el mensaje truncado junto con un aviso de dicho truncamiento.

El programa muestra cómo debe borrar los campos *MDMID* y *MDCID* de la estructura MQMD después de cada llamada MQGET porque la llamada establece estos campos en los valores contenidos en el mensaje que recupera. Si se limpian estos campos, sucesivas llamadas MQGET recuperarán los mensajes en el orden en que estén guardados en la cola.

El programa continúa hasta el final de la cola; aquí, la llamada MQGET devuelve el código de razón RC2033 (sin mensaje disponible) y el programa muestra un mensaje de aviso. Si la llamada MQGET falla, el programa escribe un mensaje de error que contiene el código de razón en su archivo de spool.

A continuación, el programa cierra la cola con la llamada MQCLOSE.

El programa de ejemplo Get en IBM i

El programa de ejemplo Get, AMQ3GET4, obtiene mensajes de una cola utilizando la llamada MQGET.

Cuando se llama al programa, elimina los mensajes de la cola especificada. Puede utilizar la cola proporcionada SYSTEM.SAMPLE.LOCAL; ejecute primero el programa de ejemplo Put para colocar algunos mensajes en la cola. Puede utilizar SYSTEM.SAMPLE.ALIAS , que es un nombre de alias para la misma cola local. El programa continúa hasta que la cola está vacía o falla una llamada MQI.

Un ejemplo de un mandato para llamar al programa RPG es:

```
CALL PGM(QMQMSAMP/AMQ3GET4) PARM('Queue_Name','Queue_Manager_Name')
```

donde `Queue_Name` y `Queue_Manager_Name` deben tener 48 caracteres de longitud, lo que se consigue rellenando `Queue_Name` y `Queue_Manager_Name` con el número necesario de espacios en blanco. Por lo tanto, si utiliza SYSTEM.SAMPLE.LOCAL como cola de destino, necesitará 29 caracteres en blanco.

Diseño del programa de ejemplo de obtención

El programa abre la cola de destino para obtener mensajes; utiliza la llamada MQOPEN con la opción OOINPQ. Si no puede abrir la cola, el programa escribe un mensaje de error que contiene el código de razón devuelto por la llamada MQOPEN en su archivo de spool.

Para cada mensaje de la cola, el programa utiliza la llamada MQGET para eliminar el mensaje de la cola; a continuación, muestra los datos contenidos en el mensaje. La llamada MQGET utiliza la opción GMWT, especificando un intervalo de espera (*GMWI*) de 15 segundos, para que el programa espere este periodo si no hay ningún mensaje en la cola. Si no llega ningún mensaje antes de que caduque este intervalo, la llamada falla y devuelve el código de razón RC2033 (no hay ningún mensaje disponible).

El programa muestra cómo debe borrar los campos *MDMID* y *MDCID* de la estructura MQMD después de cada llamada MQGET porque la llamada establece estos campos en los valores contenidos en el mensaje que recupera. Si se limpian estos campos, sucesivas llamadas MQGET recuperarán los mensajes en el orden en que estén guardados en la cola.

La llamada MQGET especifica un búfer de tamaño fijo. Si un mensaje es más largo que este búfer, la llamada falla y el programa se para.

El programa continúa hasta que la llamada MQGET devuelve el código de razón RC2033 (sin mensaje disponible) o la llamada MQGET falla. Si la llamada falla, el programa muestra un mensaje de error que contiene el código de razón.

A continuación, el programa cierra la cola con la llamada MQCLOSE.

Programa de ejemplo de solicitud en IBM i

El programa de ejemplo de solicitud, AMQ3REQ4, muestra el proceso cliente/servidor. El ejemplo es el cliente que coloca los mensajes de solicitud en una cola procesada por un programa servidor. Espera a que el programa servidor coloque un mensaje de respuesta en una cola de respuestas.

El ejemplo de solicitud coloca una serie de mensajes de solicitud en una cola utilizando la llamada MQPUT. Estos mensajes especifican SYSTEM.SAMPLE.REPLY como cola de respuesta. El programa espera mensajes de respuesta y, a continuación, los muestra. Las respuestas se envían sólo si la cola de destino (a la que llamaremos *cola de servidor*) está siendo procesado por una aplicación de servidor, o si se desencadena una aplicación para tal fin (los programas de ejemplo Inquire y Set están diseñados para ser activados). La muestra espera 5 minutos a que llegue la primera respuesta (para dar tiempo a que se desencadene una aplicación de servidor) y 15 segundos para las respuestas posteriores, pero puede finalizar sin obtener ninguna respuesta.

Para iniciar el programa, invóquelo pasándole el nombre de la cola de destino como parámetro. El programa coloca un conjunto de mensajes fijos en la cola; estos mensajes se toman del bloque de datos al final del código fuente del programa.

Diseño del programa de ejemplo de solicitud

El programa abre la cola del servidor para que pueda transferir mensajes. Utiliza la llamada MQOPEN con la opción OOOUT. Si no puede abrir la cola, el programa muestra un mensaje de error que contiene el código de razón devuelto por la llamada MQOPEN.

A continuación, el programa abre la cola de respuestas llamada SYSTEM.SAMPLE.REPLY para que se puedan obtener los mensajes de respuesta. Para ello, el programa utiliza la llamada MQOPEN con la opción OOINPX. Si no puede abrir la cola, el programa muestra un mensaje de error que contiene el código de razón devuelto por la llamada MQOPEN.

Por cada línea de entrada, el programa lee el texto en un búfer y utiliza la llamada MQPUT para crear un mensaje de solicitud que contiene el texto de dicha línea. En esta llamada, el programa utiliza la opción de informe ROEXCD para solicitar que cualquier mensaje de informe enviado sobre el mensaje de solicitud incluya los primeros 100 bytes de los datos del mensaje. El programa continúa hasta llegar al final de la entrada o hasta que falla la llamada MQPUT.

A continuación, el programa utiliza la llamada MQGET para eliminar los mensajes de respuesta de la cola y visualiza los datos contenidos en las respuestas. La llamada MQGET utiliza la opción GMWT, especificando un intervalo de espera (*GMWI*) de 5 minutos para la primera respuesta (para dar tiempo a que se desencadene una aplicación de servidor) y de 15 segundos para las respuestas posteriores. El programa espera estos periodos si no hay ningún mensaje en la cola. Si no llega ningún mensaje antes de que caduque este intervalo, la llamada falla y devuelve el código de razón RC2033 (no hay ningún

mensaje disponible). La llamada también utiliza la opción GMATM, por lo que los mensajes más largos que el tamaño de almacenamiento intermedio declarado se truncan.

El programa muestra cómo debe borrar los campos *MDMID* y *MDCOD* de la estructura MQMD después de cada llamada MQGET porque la llamada establece estos campos en los valores contenidos en el mensaje que recupera. Si se limpian estos campos, sucesivas llamadas MQGET recuperarán los mensajes en el orden en que estén guardados en la cola.

El programa continúa hasta que la llamada MQGET devuelve el código de razón RC2033 (sin mensaje disponible) o la llamada MQGET falla. Si la llamada falla, el programa muestra un mensaje de error que contiene el código de razón.

A continuación, el programa cierra tanto la cola de servidor como la cola de respuesta utilizando la llamada MQCLOSE. Tabla 813 en la página 1456 muestra los cambios en el programa de ejemplo Echo que son necesarios para ejecutar los programas de ejemplo Inquire y Set.

Nota: Los detalles del programa de ejemplo Echo se incluyen como referencia.

Tabla 813. Detalles de programa de ejemplo de cliente/servidor

Nombre de programa	Cola SYSTEM/SAMPLE	Programa iniciado
Eco	ECHO	AMQ3ECH4
Consulta	INQ	AMQ3INQ4
Conjunto	set	AMQ3SET4

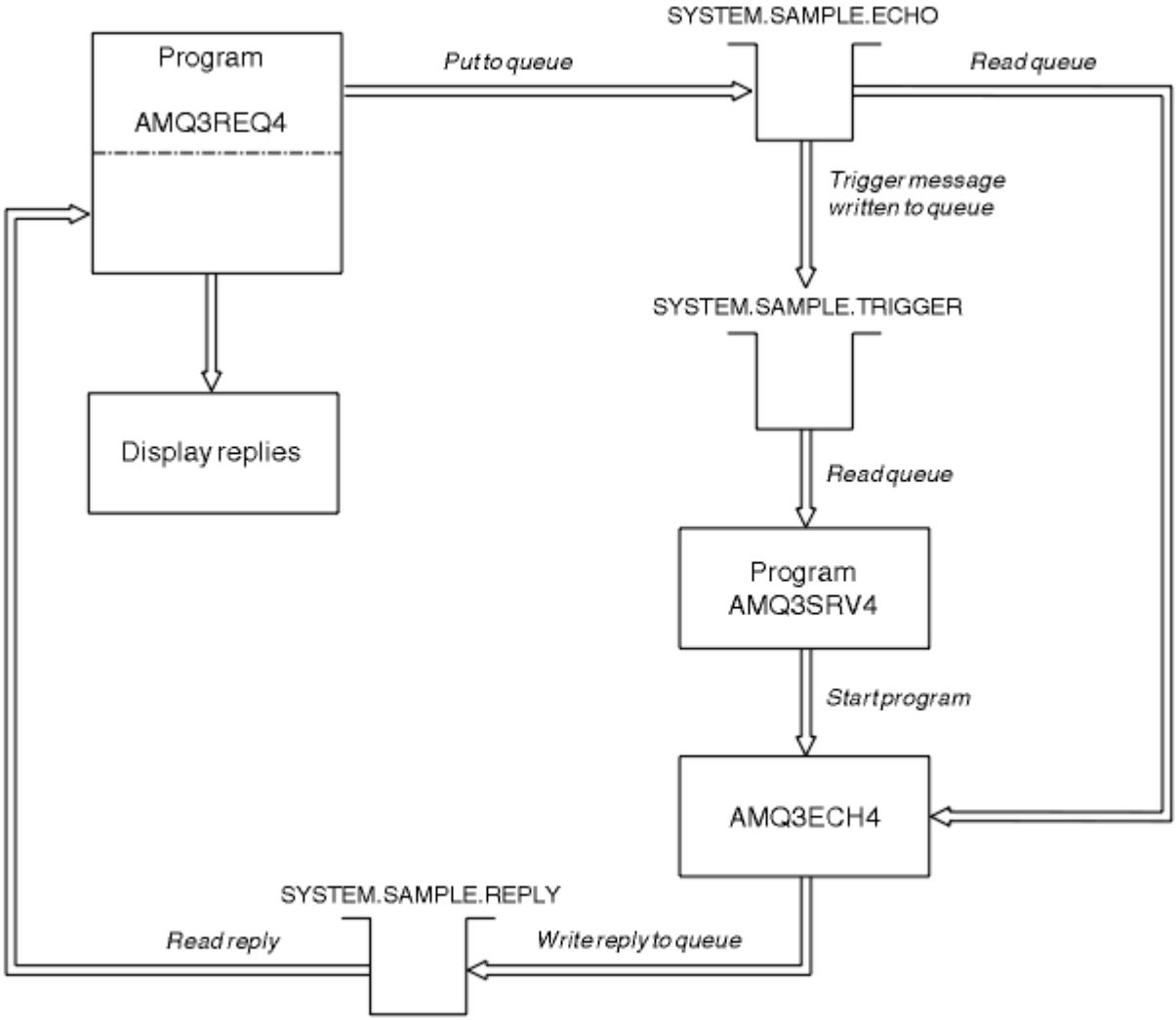


Figura 9. Diagrama de flujo del programa Cliente/Servidor (Echo) de ejemplo

IBM i *Utilización del desencadenamiento con el ejemplo de solicitud en IBM i*

Para ejecutar el ejemplo utilizando el desencadenamiento, inicie el programa servidor desencadenante, AMQ3SRV4, en la cola de inicio necesaria en un trabajo y, a continuación, inicie AMQ3REQ4 en otro trabajo.

Esto significa que el servidor desencadenante estará preparado cuando el programa de ejemplo de solicitud envía un mensaje.

Nota:

1. Los ejemplos utilizan la cola SYSTEM SAMPLE TRIGGER como cola de inicio para SYSTEM.SAMPLE.ECHO, SYSTEM.SAMPLE.INQo colas locales SYSTEM.SAMPLE.SET . De forma alternativa, puede definir su propia cola de inicio.
2. Las definiciones de ejemplo creadas por AMQSAMP4 hacen que se desencadene la versión C del ejemplo. Si desea desencadenar la versión de RPG, debe cambiar las definiciones de proceso SYSTEM.SAMPLE.ECHOPROCESS y SYSTEM.SAMPLE.INQPROCESS y SYSTEM.SAMPLE.SETPROCESS. Puede utilizar el mandato CHGMQMPRC (consulte Cambiar proceso de MQ (CHGMQMPRC) para obtener más detalles) para hacerlo, o editar y ejecutar su propia versión de AMQSAMP4.
3. Debe compilar el programa de servidor desencadenante desde el origen proporcionado en QMQMSAMP/QRPGLESRC.

En función del proceso desencadenante que desee ejecutar, se debe llamar a AMQ3REQ4 con el parámetro que especifica los mensajes de solicitud que se van a colocar en una de estas colas de servidor de ejemplo:

- SYSTEM.SAMPLE.ECHO (para los programas de ejemplo Echo)
- SYSTEM.SAMPLE.INQ (para los programas de ejemplo Inquire)
- SYSTEM.SAMPLE.SET (para los programas de ejemplo Set).

Un diagrama de flujo para SYSTEM.SAMPLE.ECHO de ECHO se muestra en Figura 9 en la página 1457. Utilizando el ejemplo, el mandato para emitir la solicitud de programa RPG a este servidor es:

```
CALL PGM(QMQMSAMP/AMQ3REQ4) PARM('SYSTEM.SAMPLE.ECHO
+ 30 blank characters','Queue_Manager_Name')
```

porque el nombre de cola y el nombre de gestor de colas deben tener 48 caracteres de longitud.

Nota: Esta cola de ejemplo tiene el tipo de desencadenante FIRST, por lo que si hay mensajes en las colas antes de ejecutar el ejemplo Request, los mensajes enviados no desencadenarán las aplicaciones de servidor.

Si desea intentar otros ejemplos, puede probar las siguientes variaciones:

- Utilice AMQ3TRG4 en lugar de AMQ3SRV4 para enviar el trabajo en su lugar, pero los posibles retrasos en el envío de trabajos podrían hacer que sea menos fácil seguir lo que está sucediendo.
- Utilice SYSTEM.SAMPLE.INQ y SYSTEM.SAMPLE.SET . Utilizando el archivo de datos de ejemplo, los mandatos para emitir las solicitudes de programa RPG a estos servidores son:

  ```
  CALL PGM(QMQMSAMP/AMQ3INQ4) PARM('SYSTEM.SAMPLE.INQ
  + 31 blank characters')
  CALL PGM(QMQMSAMP/AMQ3SET4) PARM('SYSTEM.SAMPLE.SET
  + 31 blank characters')
  ```

  porque el nombre de cola debe tener 48 caracteres de longitud.

  Estas colas de ejemplo también tienen el tipo de desencadenante FIRST.

El programa de ejemplo Echo en IBM i

Los programas de ejemplo Echo devuelven el mensaje enviado a una cola de respuestas. El programa se denomina AMQ3ECH4

Para que el proceso de desencadenamiento funcione, debe asegurarse de que el programa de ejemplo Echo que desea utilizar lo desencadenan los mensajes que llegan a la cola SYSTEM.SAMPLE.ECHO. Para ello, especifique el nombre del programa de ejemplo Echo que desea utilizar en el campo *ApplId* de la definición de proceso SYSTEM.SAMPLE.ECHOPROCESS. (Para ello, puede utilizar el mandato CHGMQMPRC, que se describe en Administración de IBM i .) La cola de ejemplo tiene un tipo de desencadenante de FIRST, por lo que si ya hay mensajes en la cola antes de ejecutar el ejemplo de solicitud, el ejemplo de eco no lo desencadenan los mensajes que envía.

Cuando haya establecido la definición correctamente, primero inicie AMQ3SRV4 en un trabajo y, a continuación, inicie AMQ3REQ4 en otro. Puede utilizar AMQ3TRG4 en lugar de AMQ3SRV4, pero los posibles retrasos en el envío de trabajos podrían hacer que sea menos fácil seguir lo que está sucediendo.

Utilice los programas de ejemplo de petición para enviar mensajes a la cola SYSTEM.SAMPLE.ECHO. Los programas de ejemplo de eco envían un mensaje de respuesta que contiene los datos del mensaje de respuesta a la cola de respuestas especificada en el mensaje de petición.

Diseño del programa de ejemplo Echo

Cuando se desencadena el programa, se conecta explícitamente al gestor de colas predeterminado utilizando la llamada MQCONN. Aunque esto no es necesario en IBM i, esto significa que puede utilizar el mismo programa en otras plataformas sin cambiar el código fuente.

A continuación, el programa abre la cola nombrada en la estructura de mensajes desencadenantes que se pasó cuando se inició. (Para que quede más claro, esta cola se llamará *cola de solicitudes*). El programa usa la llamada MQOPEN para abrir esta cola para entrada compartida.

El programa utiliza la llamada MQGET para eliminar mensajes de esta cola. Esta llamada utiliza las opciones GMATM y GMWT, con un intervalo de espera de 5 segundos. El programa comprueba el descriptor de cada mensaje para ver si es un mensaje de solicitud; si no lo es, descarta el mensaje y muestra un mensaje de advertencia.

Para cada mensaje de solicitud eliminado de la cola de solicitudes, el programa utiliza la llamada MQPUT para colocar un mensaje de respuesta en la cola de respuestas. Este mensaje contiene el contenido del mensaje de solicitud.

Cuando no quedan mensajes en la cola de solicitudes, el programa cierra esa cola y se desconecta del gestor de colas.

Este programa también puede responder a los mensajes enviados a la cola desde plataformas que no sean IBM i, aunque no se proporciona ningún ejemplo para esta situación. Para que el programa ECHO funcione, usted:

- Escriba un programa, especificando correctamente los campos *Format*, *Encoding*y *CCSID* , para enviar mensajes de solicitud de texto.

  El programa ECHO solicita al gestor de colas que realice la conversión de datos del mensaje, si es necesario.
- Especifique CONVERT (*YES) en el canal emisor de IBM MQ for IBM i , si el programa que ha escrito no proporciona una conversión similar para la respuesta.

El programa de ejemplo de consulta en IBM i

El programa de ejemplo de consulta, AMQ3INQ4, consulta algunos de los atributos de una cola utilizando la llamada MQINQ.

El programa está pensado para ejecutarse como un programa desencadenado, por lo que su única entrada es una estructura MQTMC (mensaje desencadenante). Esta estructura contiene el nombre de la cola de destino cuyos atributos se van a consultar.

Para que el proceso de desencadenamiento funcione, debe asegurarse de que el programa de ejemplo de consulta se desencadene mediante los mensajes que llegan a la cola SYSTEM.SAMPLE.INQ. Para ello, especifique el nombre del programa de ejemplo de consulta en el campo *ApplId* de SYSTEM.SAMPLE.INQPROCESS . (Para ello, puede utilizar el mandato CHGMQMPRC, descrito en Cambiar proceso de MQ (CHGMQMPRC)). La cola de ejemplo tiene un tipo de desencadenante FIRST, por lo que si ya hay mensajes en la cola antes de ejecutar el ejemplo de solicitud, los mensajes que envíe no desencadenarán el ejemplo de consulta.

Cuando haya establecido la definición correctamente, primero inicie AMQ3SRV4 en un trabajo y, a continuación, inicie AMQ3REQ4 en otro. Puede utilizar AMQ3TRG4 en lugar de AMQ3SRV4, pero los posibles retrasos en el envío de trabajos pueden hacer que sea menos fácil seguir lo que está sucediendo.

Utilice el programa de ejemplo de solicitud para enviar mensajes de solicitud, cada uno de los cuales contiene sólo un nombre de cola, a la cola SYSTEM.SAMPLE.INQ. Para cada mensaje de solicitud, el programa de ejemplo de consulta envía un mensaje de respuesta que contiene información sobre la cola especificada en el mensaje de solicitud. Las respuestas se envían a la cola de respuestas especificada en el mensaje de solicitud.

Diseño del programa de ejemplo de consulta

Cuando se desencadena el programa, se conecta explícitamente al gestor de colas predeterminado utilizando la llamada MQCONN. Aunque no es necesario en IBM i, esta característica de diseño significa que puede utilizar el mismo programa en otras plataformas sin cambiar el código fuente.

A continuación, el programa abre la cola nombrada en la estructura de mensajes desencadenantes que se pasó cuando se inició. (Para que quede más claro, esta cola se llamará *cola de solicitudes*). El programa usa la llamada MQOPEN para abrir esta cola para entrada compartida.

El programa utiliza la llamada MQGET para eliminar mensajes de esta cola. Esta llamada utiliza las opciones GMATM y GMWT, con un intervalo de espera de 5 segundos. El programa prueba el descriptor de cada mensaje para saber si es un mensaje de solicitud. Si no lo es, descarta el mensaje y visualiza un mensaje de aviso.

Para cada mensaje de solicitud eliminado de la cola de solicitudes, el programa lee el nombre de la cola (al que llamaremos la *cola de destino*) contenido en los datos y abre esa cola utilizando la llamada MQOPEN con la opción OOINQ. A continuación, el programa utiliza la llamada MQINQ para consultar los valores de los atributos **InhibitGet**, **CurrentQDepth**y **OpenInputCount** de la cola de destino.

Si la llamada MQINQ es satisfactoria, el programa utiliza la llamada MQPUT para colocar un mensaje de respuesta en la cola de respuesta. Este mensaje contiene los valores de los tres atributos.

Si la llamada MQOPEN o MQINQ no es satisfactoria, el programa utiliza la llamada MQPUT para colocar un mensaje *report* en la cola de respuesta. En el campo *MDFB* del descriptor de mensaje de este mensaje de informe se encuentra el código de razón devuelto por la llamada MQOPEN o MQINQ, en función de cuál haya fallado.

Después de la llamada MQINQ, el programa cierra la cola de destino con la llamada MQCLOSE.

Cuando no quedan mensajes en la cola de solicitudes, el programa cierra esa cola y se desconecta del gestor de colas.

El programa de ejemplo Set en IBM i

El programa de ejemplo Set, AMQ3SET4, inhibe las operaciones de colocación en una cola utilizando la llamada MQSET para cambiar el atributo **InhibitPut** de la cola.

El programa está pensado para ejecutarse como un programa desencadenado, por lo que su única entrada es una estructura MQTMC (mensaje desencadenante) que contiene el nombre de una cola de destino con atributos sobre los que se debe consultar.

Para que el proceso de desencadenamiento funcione, debe asegurarse de que el programa de ejemplo Set se desencadena mediante los mensajes que llegan a la cola SYSTEM.SAMPLE.SET. Para ello, especifique el nombre del programa de ejemplo Set en el campo *ApplId* de la definición de proceso SYSTEM.SAMPLE.SETPROCESS. (Para ello, puede utilizar el mandato CHGMQMPRC, descrito en Administración de IBM i .) La cola de ejemplo tiene un tipo de desencadenante de FIRST, por lo que si ya hay mensajes en la cola antes de ejecutar el ejemplo de solicitud, los mensajes que envíe no desencadenan el ejemplo de conjunto.

Cuando haya establecido la definición correctamente, primero inicie AMQ3SRV4 en un trabajo y, a continuación, inicie AMQ3REQ4 en otro. Puede utilizar AMQ3TRG4 en lugar de AMQ3SRV4, pero los posibles retrasos en el envío de trabajos podrían hacer que sea menos fácil seguir lo que está sucediendo.

Utilice el programa de ejemplo de solicitud para enviar mensajes de solicitud, cada uno de los cuales contiene sólo un nombre de cola, a la cola SYSTEM.SAMPLE.SET. Para cada mensaje de solicitud, el programa de ejemplo Set envía un mensaje de respuesta que contiene una confirmación de que las operaciones de transferencia se han inhibido en la cola especificada. Las respuestas se envían a la cola de respuestas especificada en el mensaje de solicitud.

Diseño del programa de ejemplo Set

Cuando se desencadena el programa, se conecta explícitamente al gestor de colas predeterminado utilizando la llamada MQCONN. Aunque no es necesario en IBM i, esto significa que puede utilizar el mismo programa en otras plataformas sin cambiar el código fuente.

A continuación, el programa abre la cola nombrada en la estructura de mensajes desencadenantes que se pasó cuando se inició. (Para que quede más claro, esta cola se llamará *cola de solicitudes*). El programa usa la llamada MQOPEN para abrir esta cola para entrada compartida.

El programa utiliza la llamada MQGET para eliminar mensajes de esta cola. Esta llamada utiliza las opciones GMATM y GMWT, con un intervalo de espera de 5 segundos. El programa comprueba el descriptor de cada mensaje para ver si es un mensaje de solicitud; si no lo es, descarta el mensaje y muestra un mensaje de advertencia.

Para cada mensaje de solicitud eliminado de la cola de solicitudes, el programa lee el nombre de la cola (al que llamaremos la *cola de destino*) contenida en los datos y abre esa cola utilizando la llamada MQOPEN con la opción OOSET. A continuación, el programa utiliza la llamada MQSET para establecer el valor del atributo **InhibitPut** de la cola de destino en QAPUTI.

Si la llamada MQSET es satisfactoria, el programa utiliza la llamada MQPUT para colocar un mensaje de respuesta en la cola de respuesta. Este mensaje contiene la serie `PUT inhibited`.

Si la llamada MQOPEN o MQSET no es satisfactoria, el programa utiliza la llamada MQPUT para colocar un mensaje *report* en la cola de respuesta. En el campo *MDFB* del descriptor de mensaje de este mensaje de informe se encuentra el código de razón devuelto por la llamada MQOPEN o MQSET, en función de cuál haya fallado.

Después de la llamada MQSET, el programa cierra la cola de destino utilizando la llamada MQCLOSE.

Cuando no quedan mensajes en la cola de solicitudes, el programa cierra esa cola y se desconecta del gestor de colas.

Los programas de ejemplo de desencadenamiento en IBM i

IBM MQ for IBM i proporciona dos programas de ejemplo de desencadenamiento escritos en ILE/RPG.

Los programas son:

AMQ3TRG4
: Se trata de un supervisor desencadenante para el entorno IBM i. Envía un trabajo IBM i para que se inicie la aplicación, pero esto significa que hay un coste de proceso adicional asociado a cada mensaje desencadenante.

AMQ3SRV4
: Se trata de un servidor desencadenante para el entorno IBM i. Por cada mensaje desencadenante, este servidor ejecuta el comando de inicio en su propio trabajo para iniciar la aplicación especificada. El servidor desencadenante puede invocar las transacciones CICS.

Las versiones en lenguaje C de estos ejemplos también están disponibles como programas ejecutables en la biblioteca QMQM, denominada AMQSTRG4 y AMQSERV4.

El supervisor desencadenante de ejemplo AMQ3TRG4 en IBM i

AMQ3TRG4 es un supervisor desencadenante. Toma un parámetro: el nombre de la cola de inicio a la que va a servir. AMQSAMP4 define un ejemplo de cola de inicio, SYSTEM.SAMPLE.TRIGGER, que se podrá utilizar al intentar ejecutar los programas de ejemplo.

AMQ3TRG4 somete un trabajo IBM i para cada mensaje desencadenante válido que obtiene de la cola de inicio.

Diseño del supervisor desencadenante

El supervisor desencadenante abre la cola de inicio y obtiene mensajes de la cola, especificando un intervalo de espera ilimitado.

El supervisor desencadenante somete un trabajo IBM i para iniciar la aplicación especificada en el mensaje desencadenante y pasa una estructura MQTMC (una versión de carácter del mensaje desencadenante). Los datos de entorno del mensaje desencadenante se utilizan como parámetros de envío de trabajos.

Por último, el programa cierra la cola de inicio.

El servidor desencadenante de ejemplo AMQ3SRV4

AMQ3SRV4 es un servidor desencadenante. Toma un parámetro: el nombre de la cola de inicio a la que va a servir. AMQSAMP4 define un ejemplo de cola de inicio, SYSTEM.SAMPLE.TRIGGER, que se podrá utilizar al intentar ejecutar los programas de ejemplo.

Para cada mensaje desencadenante, AMQ3SRV4 ejecuta un mandato de inicio en su propio trabajo para iniciar la aplicación especificada.

Al utilizar la cola desencadenante de ejemplo, el comando que se ha de emitir es:

```
CALL PGM(QMQM/AMQ3SRV4) PARM('Queue Name')
```

Donde `Queue  Name` debe tener 48 caracteres de longitud, lo que se consigue rellenando el nombre de cola con el número necesario de espacios en blanco. Por lo tanto, si utiliza SYSTEM.SAMPLE.TRIGGER como cola de destino, necesitará 28 caracteres en blanco.

Diseño del servidor desencadenante

El diseño del servidor desencadenante es similar al del supervisor desencadenante, excepto el servidor desencadenante:

- Permite aplicaciones CICS así como IBM i
- No utiliza los datos de entorno del mensaje desencadenante
- Llama a aplicaciones IBM i en su propio trabajo (o utiliza STRCICSUSR para iniciar aplicaciones CICS) en lugar de enviar un trabajo IBM i
- Abre la cola de inicio para la entrada compartida, por lo que muchos servidores desencadenantes se pueden ejecutar al mismo tiempo

Nota: Los programas iniciados por AMQ3SRV4 no deben utilizar la llamada MQDISC porque esto detendrá el servidor desencadenante. Si los programas iniciados por AMQ3SRV4 utilizan la llamada MQCONN, obtendrán el código de razón RC2002 .

Terminación de los programas de ejemplo de desencadenante en IBM i
Un programa de supervisor desencadenante puede ser finalizado por la opción 2 de sysrequest (ENDRQS) o inhibiendo las obtenciones de la cola de desencadenantes.

Si se utiliza la cola desencadenante de ejemplo, el mandato es:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*NO)
```

Nota: Para empezar a desencadenar de nuevo en esta cola, debe especificar el mandato:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Ejecución de los ejemplos utilizando colas remotas en IBM i

Puede demostrar la gestión de colas remotas ejecutando los ejemplos en gestores de colas de mensajes conectados.

El programa AMQSAMP4 proporciona una definición local de una cola remota (SYSTEM.SAMPLE.REMOTE) que utiliza un gestor de colas remoto denominado OTHER. Para utilizar esta definición de ejemplo, cambie OTHER por el nombre del segundo gestor de colas de mensajes que desee utilizar. También debe configurar un canal de mensajes entre los dos gestores de colas de mensajes; para obtener información sobre cómo hacerlo, consulte Programas de salida de canal para canales de mensajería.

El programa de ejemplo de solicitud coloca su propio nombre de gestor de colas local en el campo *MDRM* de los mensajes que envía. Los ejemplos de consulta y establecimiento envían mensajes de respuesta a la cola y al gestor de colas de mensajes especificados en los campos *MDRQ* y *MDRM* de los mensajes de solicitud que procesan.

Códigos de retorno para IBM i (ILE RPG)

Esta información describe los códigos de retorno asociados con la MQI y la MQAI.

Los códigos de retorno asociados con:

- Los mandatos PCF (Programmable Command Format) se listan en Referencia de formatos de mandato programable.
- Las llamadas C++ se listan en Utilización de C++.

Por cada llamada, el gestor de colas o una rutina de salida devuelven un código de terminación y un código de razón para indicar que la llamada se ha realizado de forma satisfactoria o bien con errores.

Las aplicaciones no deben depender de los errores que se comprueban en un orden específico, excepto cuando se indique específicamente. Si puede surgir más de un código de terminación o código de razón de una llamada, el error concreto notificado depende de la implementación.

Códigos de terminación para IBM i (ILE RPG)

El parámetro de código de terminación (*CMPCOD*) permite al interlocutor ver rápidamente si la llamada se ha completado correctamente, se ha completado parcialmente o ha fallado.

CCOK
(MQCC_OK en otras plataformas)

Realización satisfactoria.

La llamada se ha completado del todo; se han establecido todos los parámetros de salida. El parámetro **REASON** siempre tiene el valor RCNONE en este caso.

CCWARN
(MQCC_WARN en otras plataformas)

Aviso (finalización parcial).

La llamada se ha completado parcialmente. Es posible que algunos parámetros de salida se hayan establecido además de los parámetros de salida *CMPCOD* y *REASON* . El parámetro **REASON** proporciona información adicional sobre la finalización parcial.

CCFAIL
(MQCC_FAIL en otras plataformas)

La llamada no se ha realizado satisfactoriamente.

El procesamiento de la llamada no se ha completado y el estado del gestor de colas no suele variar; se indican excepciones concretas. Se han establecido los parámetros de salida *CMPCOD* y *REASON* ; otros parámetros no se modifican, excepto cuando se indique lo contrario.

La razón puede ser un error en el programa de aplicación, o puede ser un resultado de alguna situación externa al programa, por ejemplo, es posible que se haya revocado la autorización del usuario. El parámetro **REASON** proporciona información adicional sobre el error.

Códigos de razón para IBM i (ILE RPG)

El parámetro de código de razón (*REASON*) es una calificación para el parámetro de código de terminación (*CMPCOD*).

Si no hay ninguna razón especial para informar, se devuelve RCNONE. Una llamada satisfactoria devuelve CCOK y RCNONE.

Si el código de terminación es CCWARN o CCFAIL, el gestor de colas siempre informa de una razón de calificación; los detalles se proporcionan bajo cada descripción de llamada.

Allá donde las salidas de usuario establezcan códigos de terminación y de razón, habrán de atenerse a estas reglas. Además, los valores de razón especiales definidos por salidas de usuario han de ser menors que cero para garantizar que no entran en conflicto con los valores definidos por el gestor de colas. Las salidas pueden establecer razones ya definidas por el gestor de colas cuando procedan.

También aparecen códigos de razón en:

- El campo *DLREA* de la estructura MQDLH
- El campo *MDFB* de la estructura MQMD

Para obtener la lista completa de códigos de razón, consulte Códigos de terminación y razón de la API.

Para encontrar el código de razón IBM i en esa lista, elimine el "RC" de la parte frontal, por ejemplo, RC2002 pasa a ser 2002. También los códigos de terminación allí se muestran como están en otras plataformas:

Tabla 814. Nombres de código de razón en IBM i y en otras plataformas

IBM i	Otras plataformas
CCOK	MQCC_OK
CCWARN	MQCC_WARN
CCFAIL	MQCC_FAIL

Reglas para validar opciones MQI para IBM i (ILE RPG)

Este tema proporciona información sobre las situaciones que generan un código de razón RC2046 a partir de una llamada MQOPEN, MQPUT, MQPUT1, MQGET o MQCLOSE.

Llamada MQOPEN en IBM i

Para las opciones de la llamada MQOPEN:

- Se debe especificar *Al menos uno* de los siguientes:
  - OOBRW
  - OOINPQ
  - OOINPX
  - OOINPS
  - OOINQ
  - OOOUT
  - OOSET
- Solo se permite *uno* de los siguientes:
  - OOINPQ
  - OOINPX
  - OOINPS
- Solo se permite *uno* de los siguientes:
  - OOBNDO
  - OOBNDN
  - OOBNDQ

  Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, debido a que el valor de OOBNDQ es cero, si se especifica con cualquiera de las otras dos opciones de enlace no se genera el código de razón RC2046. Se proporciona OOBNDQ para ayudar a la documentación del programa.
- Si se especifica OOSAVA, también debe especificarse una de las opciones OOINP*.
- Si se especifica una de las opciones OOSET* u OOPAS*, también debe especificarse OOOUT.

Llamada MQPUT en IBM i

Para las opciones de colocación de mensajes:

- La combinación de PMSYP y PMNSYP no está permitida.
- Solo se permite *uno* de los siguientes:

  - PMDEFC
  - PNOC
  - PMPASA
  - PMPASI
  - PMSETA
  - PMSETI
- PMALTU no está permitido (sólo es válido en la llamada MQPUT1).

Llamada MQPUT1 en IBM i

Para las opciones de transferencia de mensajes, las reglas son las mismas que para la llamada MQPUT, excepto para las opciones siguientes:

- PMALTU está permitido.
- PMLOGO no está permitido.

Llamada MQGET en IBM i

Para las opciones get-message:

- Solo se permite *una* de las opciones siguientes:
  - GMNSYP
  - GMSYP
  - GMPSYP
- Solo se permite *una* de las opciones siguientes:
  - GMBRWF
  - GMBRWC
  - GMBRWN
  - GMMUC
- GMSYP no está permitido con ninguna de las opciones siguientes:
  - GMBRWF
  - GMBRWC
  - GMBRWN
  - GMLK
  - GMUNLK
- GMPSYP no está permitido con ninguna de las opciones siguientes:
  - GMBRWF
  - GMBRWC
  - GMBRWN
  - GMCMPM
  - GMUNLK
- Si se especifica GMLK, también debe especificarse una de las opciones siguientes:
  - GMBRWF
  - GMBRWC
  - GMBRWN
- Si se especifica GMUNLK, solo se permiten las opciones siguientes:

  - GMNSYP
  - GMNWT

Llamada MQCLOSE en IBM i

- Para las opciones de la llamada MQCLOSE. La combinación de CODEL y COPURG no está permitida.
- Sólo se permite uno de los siguientes:
  - COKPSB
  - CORMSB

Llamada MQSUB en IBM i

Para las opciones de la llamada MQSUB:

- Se debe especificar al menos uno de los siguientes:
- Se debe especificar al menos uno de los siguientes:
  - SOALT
  - SORES
  - SOCRT
- Sólo se permite uno de los siguientes:
  - SODUR
  - SONDUR

  Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de SOnDUR es cero, si se especifica con SODUR no se genera el código de razón RC2046. SONDUR se proporciona para ayudar a la documentación del programa.
- La combinación de SOGRP y SOMAN no está permitida.
- SOGRP requiere que se especifique SOSCID.
- Solo se permite uno de los siguientes: SOAUID SOFUID
- La combinación de SONEWP y SOPUBR no está permitida.
- SONEWP solo está permitido en combinación con SOCRT.
- Sólo se permite uno de los siguientes:
  - SOCHR
  - SOWTOP

Codificaciones de máquina en IBM i

Utilice esta información para obtener información sobre la estructura del campo *MDENC* en el descriptor de mensaje.

Para obtener más información sobre el descriptor de mensaje, consulte “MQMD (Descriptor de mensaje) en IBM i” en la página 1133.

El campo *MDENC* es un entero de 32 bits que se divide en cuatro subcampos separados; estos subcampos identifican:

- La codificación utilizada para enteros binarios
- La codificación utilizada para enteros de decimal empaquetado
- La codificación utilizada para números de coma flotante
- Bits reservados

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Se definen las máscaras siguientes:

ENIMSK
Máscara para codificación de enteros binarios.

Este subcampo ocupa las posiciones de bits 28 a 31 dentro del campo *MDENC* .

ENDMSK
Máscara para codificación de enteros decimales empaquetados.

Este subcampo ocupa las posiciones de bits 24 a 27 dentro del campo *MDENC* .

ENFMSK
Máscara para codificación de coma flotante.

Este subcampo ocupa las posiciones de bits 20 a 23 dentro del campo *MDENC* .

ENRMSK
Máscara para bits reservados.

Este subcampo ocupa las posiciones de bits 0 a 19 dentro del campo *MDENC* .

IBM i Codificación de enteros binarios en IBM i

Valores válidos para la codificación de enteros binarios.

Los valores siguientes son válidos para la codificación de entero binario:

ENIUND
Codificación de enteros no definida.

Los enteros binarios se representan utilizando una codificación que no está definida.

ENINOR
Codificación de enteros normal.

Los enteros binarios se representan de la forma convencional:

- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior.

ENIREV
Codificación de enteros invertida.

Los enteros binarios se representan de la misma forma que ENINOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENINOR.

IBM i Codificación de enteros decimales empaquetados en IBM i

Valores válidos para la codificación de enteros decimales empaquetados

Los valores siguientes son válidos para la codificación de enteros decimales empaquetados:

ENDUND
Codificación de decimal empaquetado no definida.

Los enteros de decimal empaquetado se representan utilizando una codificación que no está definida.

ENDNOR
Codificación de decimal empaquetado normal.

Los enteros decimales empaquetados se representan de la forma convencional:

- Cada dígito decimal en la forma imprimible del número se representa en decimal empaquetado por un dígito hexadecimal único en el rango de X' 0 'a X' 9'. Cada dígito hexadecimal ocupa 4 bits, por

lo que cada byte del número decimal empaquetado representa dos dígitos decimales en el formato imprimible del número.

- El byte menos significativo en el número decimal empaquetado es el byte que contiene el dígito decimal menos significativo. Dentro de ese byte, los 4 bits más significativos contienen el dígito decimal menos significativo, y los 4 bits menos significativos contienen el signo. El signo es X'C '(positivo), X 'D' (negativo) o X'F ' (sin signo).
- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior.

ENDREV
Codificación de decimal empaquetado invertida.

Los enteros decimales empaquetados se representan de la misma forma que ENDNOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENDNOR.

IBM i Codificación de coma flotante en IBM i

Valores válidos para la codificación de coma flotante

Los valores siguientes son válidos para la codificación de coma flotante:

ENFUND
Codificación de coma flotante no definida.

Los números de coma flotante se representan utilizando una codificación que no está definida.

ENFNOR
Codificación normal de flotador IEEE (Instituto de Ingenieros Eléctricos y Electrónicos).

Los números de coma flotante se representan utilizando el formato de coma flotante IEEE estándar, con los bytes ordenados de la siguiente manera:

- El byte menos significativo de la mantisa tiene la dirección más alta de cualquiera de los bytes del número; el byte que contiene el exponente tiene la dirección más baja
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior

Los detalles de la codificación flotante IEEE pueden encontrarse en el estándar IEEE 754.

ENFREV
Codificación flotante IEEE invertida.

Los números de coma flotante se representan de la misma forma que ENFNOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENFNOR.

ENF390
Codificación flotante de la arquitectura System/390 .

Los números de coma flotante se representan utilizando el formato de coma flotante estándar System/390 ; también lo utiliza System/370.

IBM i Construcción de codificaciones en IBM i

Para construir un valor para el campo *MDENC* en MQMD, se deben añadir las constantes relevantes que describen las codificaciones necesarias.

Asegúrese de combinar sólo una de las codificaciones ENI* con una de las codificaciones END* y una de las codificaciones ENF*.

IBM i Análisis de codificaciones en IBM i

El campo *MDENC* contiene subcampos; debido a esto, las aplicaciones que necesitan examinar la codificación entera, decimal empaquetado o flotante deben utilizar la técnica descrita en este tema.

Utilización aritmética

Los pasos siguientes se deben realizar utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
   - 1 para la codificación de enteros binarios
   - 16 para la codificación de enteros decimales empaquetados
   - 256 para la codificación de coma flotante

   Llame al valor A.
2. Divida el valor del campo *MDENC* por A ; llamar al resultado B.
3. Divida B por 16; llame al resultado C.
4. Multiplique C por 16 y reste de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. E es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

IBM i Resumen de codificaciones de arquitectura de máquina en IBM i

Una tabla que resume las codificaciones para arquitecturas de máquina.

Las codificaciones para arquitecturas de máquina se muestran en Tabla 815 en la página 1469.

Tabla 815. Resumen de codificaciones para arquitecturas de máquina

Arquitectura de máquina	Codificación de enteros binarios	Codificación de enteros de decimal empaquetado	Codificación de coma flotante
IBM i	normal	normal	IEEE normal
Intel x86	reversed	reversed	IEEE invertido
PowerPC	normal	normal	IEEE normal
System/390	normal	normal	System/390

IBM i Opciones de informe y distintivos de mensaje en IBM i

Este tema se refiere a los campos *MDREP* y *MDMFL* que forman parte del descriptor de mensaje MQMD especificado en las llamadas MQGET, MQPUT y MQPUT1 .

Para obtener más información sobre el descriptor de mensaje, consulte “MQMD (Descriptor de mensaje) en IBM i” en la página 1133. Esta información describe:

- La estructura del campo de informe y cómo lo procesa el gestor de colas
- Cómo debe analizar una aplicación el campo de informe
- La estructura del campo de distintivos de mensaje

Estructura del campo de informe

El campo *MDREP* es un entero de 32 bits que se divide en tres subcampos separados.

Estos subcampos identifican:

- Opciones de informe que se rechazan si el gestor de colas local no las reconoce
- Opciones de informe que siempre se aceptan, incluso si el gestor de colas local no las reconoce
- Opciones de informe que sólo se aceptan si se cumplen otras condiciones

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Tenga en cuenta que los bits de un subcampo no son necesariamente adyacentes. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

RORO
Máscara para opciones de informe no soportadas que se rechazan.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local harán que la llamada MQPUT o MQPUT1 falle con el código de terminación CCFAIL y el código de razón RC2061.

Este subcampo ocupa las posiciones de bits 3 y 11 a 13.

ROAUM
Enmascarar las opciones de informe no soportadas que se aceptan.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 . En este caso, se devuelve el código de terminación CCWARN con el código de razón RC2104 .

Este subcampo ocupa las posiciones de bits 0 a 2, 4 a 10 y 24 a 31.

En este subcampo se incluyen las siguientes opciones de informe:

- RCMTC
- RODLQ
- DISCO
- ROEXC
- ROEXCD
- ROEXCF
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- RONONA
- ROPAN
- ROPCI
- ROPMI

ROAUXM
Enmascarar las opciones de informe no soportadas que sólo se aceptan en determinadas circunstancias.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ODMN* y *ODON* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

El código de terminación CCWARN con el código de razón RC2104 se devuelve si se cumplen estas condiciones y CCFAIL con el código de razón RC2061 si no es así.

Este subcampo ocupa las posiciones de bits 14 a 23.

En este subcampo se incluyen las siguientes opciones de informe:

- ROCOA
- ROCOAD
- ROCOAF
- ROCOD
- ROCODD
- ROCODF

Si hay alguna opción especificada en el campo *MDREP* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MDREP* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Si se devuelve CCWARN, no se define qué código de razón se devuelve si existen otras condiciones de aviso.

La posibilidad de especificar y tener opciones de informe aceptadas que no son reconocidas por el gestor de colas local es útil cuando es necesario enviar un mensaje con una opción de informe que será reconocida y procesada por un gestor de colas *remoto* .

IBM i Análisis del campo de informe en IBM i

El campo MDREP contiene subcampos. Debido a esto, algunas aplicaciones necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado. Estas aplicaciones deben utilizar la técnica descrita en este tema.

Utilización aritmética

Los pasos siguientes se deben realizar utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:
   - Informe ROCOA para COA
   - Informe ROCOD para COD
   - ROEXC para informe de excepción
   - ROEXP para informe de caducidad

   Llame al valor A.
2. Divida el campo *MDREP* por A ; llamar al resultado B.
3. Divida B por 8 ; llamar al resultado C.
4. Multiplique C por 8 y resta de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. Pruebe E para la igualdad con cada uno de los valores posibles para ese tipo de informe.

   Por ejemplo, si A es ROEXC, pruebe la igualdad de E con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:
   - RONONA
   - ROEXC
   - ROEXCD
   - ROEXCF

Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

El pseudocódigo siguiente ilustra esta técnica para los mensajes de informe de excepción:

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Se puede utilizar un método similar para probar las opciones ROPMI o ROPCI; seleccione como valor A cualquiera de estas dos constantes y, a continuación, continúe como se ha descrito anteriormente, pero sustituyendo el valor 8 en los pasos anteriores por el valor 2.

IBM i Estructura del campo de distintivos de mensaje en IBM i

El campo *MDMFL* es un entero de 32 bits que se divide en tres subcampos separados.

Estos subcampos identifican:

- Distintivos de mensaje que se rechazan si el gestor de colas local no los reconoce
- Distintivos de mensajes que siempre se aceptan, incluso si el gestor de colas local no los reconoce
- Distintivos de mensaje que se aceptan sólo si se cumplen otras condiciones

Nota: Todos los subcampos de *MDMFL* están reservados para que los utilice el gestor de colas.

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MFRO

Máscara para distintivos de mensaje no soportados que se rechazan.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje que no están soportados por el gestor de colas local harán que la llamada MQPUT o MQPUT1 falle con el código de terminación CCFAIL y el código de razón RC2249.

Este subcampo ocupa las posiciones de bits 20 a 31.

En este subcampo se incluyen los distintivos de mensaje siguientes:

- MFLMIG
- MFLSEG
- MFMIG
- MFSEG
- MFSEGA
- MFSEGI

MFAUM

Máscara para distintivos de mensaje no soportados que se aceptan.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 . El código de terminación es CCOK.

Este subcampo ocupa las posiciones de bits 0 a 11.

MFAUXM

Máscara para distintivos de mensaje no soportados que sólo se aceptan en determinadas circunstancias.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje no soportados por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ODMN* y *ODON* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación CCOK si se cumplen estas condiciones y CCFAIL con el código de razón RC2249 si no es así.

Este subcampo ocupa las posiciones de bits 12 a 19.

Si hay distintivos especificados en el campo *MDMFL* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MDMFL* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

IBM i Conversión de datos en IBM i

En este tema se describe la interfaz para la salida de conversión de datos y el proceso realizado por el gestor de colas cuando es necesaria la conversión de datos.

La salida de conversión de datos se invoca como parte del proceso de la llamada MQGET. Se utiliza para convertir los datos de mensaje de aplicación a la representación que necesita la aplicación receptora. La conversión de los datos del mensaje de aplicación es opcional y requiere que se especifique la opción GMCONV en la llamada MQGET.

Se describen los siguientes aspectos de la conversión de datos:

- El proceso realizado por el gestor de colas en respuesta a la opción GMCONV; consulte “Proceso de conversión en IBM i” en la página 1473.
- Convenios de proceso utilizados por el gestor de colas al procesar un formato incorporado; estos convenios también se recomiendan para las salidas escritas por el usuario. Consulte “Convenios de proceso en IBM i” en la página 1475.
- Consideraciones especiales para la conversión de mensajes de informe; consulte “Conversión de mensajes de informe en IBM i” en la página 1479.
- Los parámetros pasados a la salida de conversión de datos; consulte “MQCONVX (Salida de conversión de datos) en IBM i” en la página 1490.
- Llamada que se puede utilizar desde la salida para convertir datos de tipo carácter entre distintas representaciones; consulte “MQXCNVC (Convertir caracteres) en IBM i” en la página 1485.
- El parámetro data-structure que es específico de la salida; consulte “MQDXP (parámetro de salida de conversión de datos) en IBM i” en la página 1480.

IBM i Proceso de conversión en IBM i

Esta información describe el proceso realizado por el gestor de colas en respuesta a la opción GMCONV.

El gestor de colas realiza las acciones siguientes si se especifica la opción GMCONV en la llamada MQGET y hay un mensaje que se debe devolver a la aplicación:

1. Si se cumple una o más de las condiciones siguientes, no es necesaria ninguna conversión:
   - Los datos del mensaje ya están en el juego de caracteres y la codificación que necesita la aplicación que emite la llamada MQGET. La aplicación debe establecer los campos *MDCSI* y *MDENC* en el parámetro **MSGDSC** de la llamada MQGET en los valores necesarios, antes de emitir la llamada.
   - La longitud de los datos del mensaje es cero.
   - La longitud del parámetro **BUFFER** de la llamada MQGET es cero.

En estos casos, el mensaje se devuelve sin conversión a la aplicación que emite la llamada MQGET; los valores *MDCSI* y *MDENC* del parámetro **MSGDSC** se establecen en los valores de la información de control del mensaje y la llamada se completa con una de las siguientes combinaciones de código de terminación y código de razón:

Código de terminación
Código de razón

CCOK
RCNONE

CCWARN
RC2079

CCWARN
RC2080

Los pasos siguientes sólo se realizan si el juego de caracteres o la codificación de los datos del mensaje difiere del valor correspondiente en el parámetro **MSGDSC** y hay datos que se deben convertir:

1. Si el campo *MDFMT* de la información de control del mensaje tiene el valor FMNONE, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2110.

   En todos los demás casos, el proceso de conversión continúa.
2. El mensaje se elimina de la cola y se coloca en un almacenamiento intermedio temporal que tiene el mismo tamaño que el parámetro **BUFFER** . Para las operaciones de examen, el mensaje se copia en el almacenamiento intermedio temporal, en lugar de eliminarse de la cola.
3. Si el mensaje debe truncarse para que quepa en el almacenamiento intermedio, se realiza lo siguiente:
   - Si no se ha especificado la opción GMATM, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2080.
   - Si se ha especificado la opción de GMATM , el código de terminación se establece en CCWARN, el código de razón se establece en RC2079y el proceso de conversión continúa.
4. Si el mensaje se puede acomodar en el almacenamiento intermedio sin truncamiento, o se ha especificado la opción GMATM, se realiza lo siguiente:
   - Si el formato es un formato incorporado, el almacenamiento intermedio se pasa al servicio de conversión de datos del gestor de colas.
   - Si el formato no es un formato incorporado, el almacenamiento intermedio se pasa a una salida escrita por el usuario que tiene el mismo nombre que el formato. Si no se puede encontrar la salida, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2110.

   Si no se produce ningún error, la salida del servicio de conversión de datos o de la salida escrita por el usuario es el mensaje convertido, más el código de terminación y el código de razón que se devolverá a la aplicación que emite la llamada MQGET.
5. Si la conversión es satisfactoria, el gestor de colas devuelve el mensaje convertido a la aplicación. En este caso, el código de terminación y el código de razón devueltos por la llamada MQGET normalmente serán una de las combinaciones siguientes:

   Código de terminación
   Código de razón

   CCOK
   RCNONE

   CCWARN
   RC2079

   Sin embargo, si la conversión la realiza una salida escrita por el usuario, se pueden devolver otros códigos de razón, incluso cuando la conversión es satisfactoria.

Si la conversión falla (por cualquier motivo), el gestor de colas devuelve el mensaje sin convertir a la aplicación, con los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** establecidos en los valores de la información de control del mensaje y con el código de terminación CCWARN.

IBM i Convenios de proceso en IBM i

Al convertir un formato incorporado, el gestor de colas sigue los convenios de proceso descritos en este tema.

Considere la posibilidad de aplicar estos convenios a las salidas escritas por el usuario, aunque el gestor de colas no lo aplique. Los formatos incorporados convertidos por el gestor de colas son los siguientes:

- FMADMN
- FMMDE
- FMCICS
- FMPCF
- FMCMD1
- HMRM
- FMCMD2
- FMRFH
- FMDLH
- FMRFH2
- FMDH
- FMSTR
- FMEVNT
- FMTM
- FMIMS
- FMXQH
- FMIMVS

1. Si el mensaje se expande durante la conversión y supera el tamaño del parámetro **BUFFER** , se realiza lo siguiente:
   - Si no se ha especificado la opción GMATM, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2120.
   - Si se ha especificado la opción de GMATM , el mensaje se trunca, el código de terminación se establece en CCWARN, el código de razón se establece en RC2079y el proceso de conversión continúa.
2. Si se produce un truncamiento (antes o durante la conversión), es posible que el número de bytes válidos devueltos en el parámetro **BUFFER** sea *menor que* la longitud del almacenamiento intermedio.

   Esto puede ocurrir, por ejemplo, si un entero de 4 bytes o un carácter DBCS se encuentra entre los extremos del almacenamiento intermedio. El elemento de información incompleto no se convierte y, por lo tanto, los bytes del mensaje devuelto no contienen información válida. Esto también puede ocurrir si un mensaje que se ha truncado antes de la conversión se reduce durante la conversión.

   Si el número de bytes válidos devueltos es menor que la longitud del almacenamiento intermedio, los bytes no utilizados al final del almacenamiento intermedio se establecen en nulos.
3. Si una matriz o serie se encuentra entre el final del almacenamiento intermedio, se convierte la mayor parte posible de los datos; sólo el elemento de matriz o carácter DBCS concreto que está incompleto no se convierte-se convierten los elementos o caracteres de matriz anteriores.
4. Si se produce un truncamiento (antes o durante la conversión), la longitud devuelta para el parámetro **DATLEN** es la longitud del mensaje *unconvert* antes del truncamiento.

5. Cuando las series se convierten entre juegos de caracteres de un solo byte (SBCS), juegos de caracteres de doble byte (DBCS) o juegos de caracteres de varios bytes (MBCS), las series pueden expandirse o contraerse.
   - En los formatos PCF FMADMN, FMEVNT y FMPCF, las series de las estructuras MQCFST y MQCFSL se expanden o se contraen según sea necesario para acomodar la serie después de la conversión.

     Para la estructura de lista de series MQCFSL, las series de la lista pueden expandirse o contraerse en cantidades diferentes. Si esto sucede, el gestor de colas rellena las series más cortas con espacios en blanco para que tengan la misma longitud que la serie más larga después de la conversión.
   - En el formato FMRMH, las series a las que se dirigen los campos RMSEO, RMSNO, RMDEOy RMDNO se expanden o contraen según sea necesario para acomodar las series después de la conversión.
   - En el formato FMRFH, el campo RFNVS se expande o contrae según sea necesario para acomodar los pares nombre-valor después de la conversión.
   - En estructuras con tamaños de campo fijos, el gestor de colas permite que las series se expandan o se contraigan dentro de sus campos fijos, si no se pierde información significativa. En este sentido, los espacios en blanco finales y los caracteres que siguen al primer carácter nulo del campo se tratan como insignificantes.
     - Si la serie se expande, pero sólo es necesario descartar los caracteres insignificantes para acomodar la serie convertida en el campo, la conversión se realiza correctamente y la llamada se completa con CCOK y el código de razón RCNONE (suponiendo que no haya otros errores).
     - Si la serie se expande, pero la serie convertida requiere que se descarten caracteres significativos para que quepan en el campo, el mensaje se devuelve sin convertir y la llamada se completa con CCWARN y el código de razón RC2190.

       Nota: El código de razón RC2190 da como resultado en este caso si se ha especificado la opción GMATM.
     - Si la serie se contrae, el gestor de colas rellena la serie con espacios en blanco hasta la longitud del campo.
6. Para los mensajes que constan de una o más estructuras de cabecera IBM MQ seguidas de datos de usuario, es posible convertir una o más de las estructuras de cabecera, mientras que el resto del mensaje no lo es. Sin embargo, con dos excepciones, los campos MDCSI y MDENC de cada estructura de cabecera siempre indican correctamente el juego de caracteres y la codificación de los datos que siguen a la estructura de cabecera.

   Las dos excepciones son las estructuras MQCIH y MQIIH, donde los valores de los campos MDCSI y MDENC de esas estructuras no son significativos. Para esas estructuras, los datos que siguen a la estructura están en el mismo juego de caracteres y codificación que la propia estructura MQCIH o MQIIH.
7. Si los campos MDCSI o MDENC de la información de control del mensaje que se está recuperando, o en el parámetro **MSGDSC** , especifican valores que no están definidos o no están soportados, el gestor de colas puede ignorar el error si no es necesario utilizar el valor no definido o no soportado al convertir el mensaje.

   Por ejemplo, si el campo MDENC del mensaje especifica una codificación flotante no soportada, pero el mensaje contiene sólo datos enteros, o contiene datos de coma flotante que no requieren conversión (porque las codificaciones flotantes de origen y destino son idénticas), el error puede o no diagnosticarse.

   Si se diagnostica el error, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y uno de los códigos de razón RC2111, RC2112, RC2113, RC2114 o RC2115, RC2116, RC2117, RC2118 (según corresponda); los campos MDCSI y MDENC del parámetro **MSGDSC** se establecen en los valores de la información de control del mensaje.

   Si el error no se diagnostica y la conversión se completa correctamente, los valores devueltos en los campos MDCSI y MDENC en el parámetro **MSGDSC** son los especificados por la aplicación que emite la llamada MQGET.

8. En todos los casos, si el mensaje se devuelve a la aplicación sin convertir, el código de finalización se establece en CCWARN, y los campos `MDCSI` y `MDENC` del parámetro **MSGDSC** se establecen en los valores adecuados para los datos sin convertir. Esto también se hace para FMNONE.

   El parámetro **REASON** se establece en un código que indica por qué no se ha podido llevar a cabo la conversión, a menos que el mensaje también se haya tenido que truncar; los códigos de razón relacionados con el truncamiento tienen prioridad sobre los códigos de razón relacionados con la conversión. (Para determinar si se ha convertido un mensaje truncado, compruebe los valores devueltos en los campos `MDCSI` y `MDENC` en el parámetro **MSGDSC** .)

   Cuando se diagnostica un error, se devuelve un código de razón específico o el código de razón general RC2119. El código de razón devuelto depende de las prestaciones de diagnóstico del servicio de conversión de datos subyacente.
9. Si se devuelve el código de terminación CCWARN y hay más de un código de razón relevante, el orden de prioridad es el siguiente:
   a. La razón siguiente tiene prioridad sobre todas las demás:
      - RC2079
   b. La siguiente en prioridad es la siguiente razón:
      - RC2110
   c. El orden de prioridad dentro de los códigos de razón restantes no está definido.
10. Al finalizar la llamada MQGET:
    - El siguiente código de razón indica que el mensaje se ha convertido correctamente:
      - RCNONE
    - El siguiente código de razón indica que el mensaje *puede* haberse convertido correctamente (compruebe los campos `MDCSI` y `MDENC` en el parámetro **MSGDSC** para averiguarlo):
      - RC2079
    - Los demás códigos de razón indican que el mensaje no se ha convertido.

El siguiente proceso es específico de los formatos incorporados; no es aplicable a los formatos definidos por el usuario:

1. Excepto para los formatos siguientes:
   - FMADMN
   - FMEVNT
   - FMIMVS
   - FMPCF
   - FMSTR

   ninguno de los formatos incorporados se puede convertir de o a conjuntos de caracteres que no tengan caracteres SBCS para los caracteres que son válidos en los nombres de cola. Si se intenta realizar una conversión de este tipo, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2111 o RC2115, según corresponda.

   El juego de caracteres Unicode UTF-16 es un ejemplo de un juego de caracteres que no tiene caracteres SBCS para los caracteres que son válidos en los nombres de cola.
2. Si los datos del mensaje para un formato incorporado se truncan, los campos dentro del mensaje que contienen longitudes de series, o recuentos de elementos o estructuras, no se ajustan para reflejar la longitud de los datos devueltos a la aplicación; los valores devueltos para dichos campos dentro de los datos del mensaje son los valores aplicables al mensaje antes del truncamiento.

   Al procesar mensajes como, por ejemplo, un mensaje FMADMN truncado, debe asegurarse de que la aplicación no intenta acceder a los datos más allá del final de los datos devueltos.
3. Si el nombre de formato es FMDLH, los datos de mensaje empiezan por una estructura MQDLH, y esto puede ir seguido de cero o más bytes de datos de mensaje de aplicación. El formato, el juego de

caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos DLFMT, DLCSIy DLENC en la estructura MQDLH al principio del mensaje. Puesto que la estructura MQDLH y los datos del mensaje de aplicación pueden tener diferentes conjuntos de caracteres y codificaciones, es posible que uno, otro o ambos datos de la estructura MQDLH y del mensaje de aplicación requieran conversión.

El gestor de colas convierte primero la estructura MQDLH, según sea necesario. Si la conversión es satisfactoria, o la estructura MQDLH no requiere conversión, el gestor de colas comprueba los campos DLCSI y DLENC de la estructura MQDLH para ver si es necesaria la conversión de los datos del mensaje de aplicación. Si la conversión es necesaria, el gestor de colas invoca la salida escrita por el usuario con el nombre proporcionado por el campo DLFMT en la estructura MQDLH, o realiza la propia conversión (si DLFMT es el nombre de un formato incorporado).

Si la llamada MQGET devuelve un código de terminación de CCWARN, y el código de razón es uno de los que indica que la conversión no ha sido satisfactoria, se aplica uno de los siguientes:

- No se ha podido convertir la estructura MQDLH. En este caso, los datos del mensaje de aplicación tampoco se habrán convertido.
- La estructura MQDLH se ha convertido, pero los datos del mensaje de aplicación no.

La aplicación puede examinar los valores devueltos en los campos MDCSI y MDENC en el parámetro **MSGDSC** , y los de la estructura MQDLH, para determinar cuál de los anteriores se aplica.

4. Si el nombre de formato es FMXQH, los datos del mensaje empiezan con una estructura MQXQH, y esto puede ir seguido de cero o más bytes de datos adicionales. Estos datos adicionales son normalmente los datos de mensaje de aplicación (que pueden tener una longitud cero), pero también puede haber una o más estructuras de cabecera IBM MQ adicionales presentes, al principio de los datos adicionales.

   La estructura MQXQH debe estar en el juego de caracteres y la codificación del gestor de colas. El formato, el juego de caracteres y la codificación de los datos que siguen a la estructura MQXQH se proporcionan mediante los campos MDFMT, MDCSIy MDENC en la estructura MQMD contenida en MQXQH. Para cada estructura de cabecera IBM MQ posterior presente, los campos MDFMT, MDCSIy MDENC de la estructura describen los datos que siguen a esa estructura; esos datos son otra estructura de cabecera IBM MQ o los datos de mensaje de aplicación.

   Si se especifica la opción GMCONV para un mensaje FMXQH, los datos del mensaje de aplicación y algunas de las estructuras de cabecera de MQ se convierten, pero los datos de la estructura MQXQH no. Al volver de la llamada MQGET, por lo tanto:

   - Los valores de los campos MDFMT, MDCSIy MDENC del parámetro **MSGDSC** , describen los datos de la estructura MQXQH y no los datos del mensaje de aplicación; por lo tanto, los valores no serán los mismos que los especificados por la aplicación que ha emitido la llamada MQGET.

     El efecto de esto es que una aplicación que obtiene repetidamente mensajes de una cola de transmisión con la opción GMCONV especificada debe restablecer los campos MDCSI y MDENC en el parámetro **MSGDSC** a los valores necesarios para los datos de mensaje de aplicación, antes de cada llamada MQGET.
   - Los valores de los campos MDFMT, MDCSIy MDENC de la última estructura de cabecera de MQ presente describen los datos del mensaje de aplicación. Si no hay otras estructuras de cabecera IBM MQ presentes, los datos de mensaje de aplicación se describen mediante estos campos en la estructura MQMD dentro de la estructura MQXQH. Si la conversión es satisfactoria, los valores serán los mismos que los especificados en el parámetro **MSGDSC** por la aplicación que ha emitido la llamada MQGET.

   Si el mensaje es un mensaje de lista de distribución, la estructura MQXQH va seguida de una estructura MQDH (más sus matrices de registros MQOR y MQPMR), que a su vez puede ir seguida de cero o más estructuras de cabecera IBM MQ adicionales y de cero o más bytes de datos de mensaje de aplicación. Al igual que la estructura MQXQH, la estructura MQDH debe estar en el juego de caracteres y la codificación del gestor de colas, y no se convierte en la llamada MQGET, incluso si se especifica la opción GMCONV.

El proceso de las estructuras MQXQH y MQDH descritas anteriormente están pensadas principalmente para que las utilicen los agentes de canal de mensajes cuando obtienen mensajes de las colas de transmisión.

IBM i Conversión de mensajes de informe en IBM i

Un mensaje de informe puede contener cantidades variables de datos de mensaje de aplicación, según las opciones de informe especificadas por el remitente del mensaje original.

En concreto, un mensaje de informe puede contener:

1. No hay datos de mensaje de aplicación
2. Algunos de los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica RO* D y el mensaje tiene más de 100 bytes.
3. Todos los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica RO* F, o especifica RO* D y el mensaje tiene 100 bytes o menos.

Cuando el gestor de colas o el agente de canal de mensajes genera un mensaje de informe, copia el nombre de formato del mensaje original en el campo *MDFMT* de la información de control del mensaje de informe. Por lo tanto, el nombre de formato del mensaje de informe puede implicar una longitud de datos diferente de la longitud presente en el mensaje de informe (casos 1 y 2 descritos anteriormente).

Si se especifica la opción GMCONV cuando se recupera el mensaje de informe:

- Para el caso 1 descrito anteriormente, no se invocará la salida de conversión de datos (porque el mensaje de informe no tendrá datos).
- Para el caso 3 descrito anteriormente, el nombre de formato implica correctamente la longitud de los datos del mensaje.
- Pero para el caso 2 descrito anteriormente, se invocará la salida de conversión de datos para convertir un mensaje que sea *más corto* que la longitud implícita por el nombre de formato.

  Además, el código de razón pasado a la salida será normalmente RCNONE (es decir, el código de razón no indicará que el mensaje se ha truncado). Esto sucede porque los datos del mensaje han sido truncados por el *emisor* del mensaje de informe y no por el gestor de colas del receptor en respuesta a la llamada MQGET.

Debido a estas posibilidades, la salida de conversión de datos no debe utilizar el nombre de formato para deducir la longitud de los datos que se le pasan; en su lugar, la salida debe comprobar la longitud de los datos proporcionados y estar preparada para convertir menos datos que la longitud implícita en el nombre de formato. Si los datos se pueden convertir correctamente, la salida debe devolver el código de terminación CCOK y el código de razón RCNONE. La longitud de los datos de mensaje que se van a convertir se pasa a la salida como parámetro **INLEN** .

Interfaz de programación sensible al producto

Si un mensaje de informe contiene información sobre una actividad que ha tenido lugar, se conoce como informe de actividad. Ejemplos de actividades son:

- un MCA que envía un mensaje desde una cola hacia abajo en un canal
- un MCA que recibe un mensaje de un canal y lo coloca en una cola
- un mensaje no entregado MCA que pone en cola un mensaje no entregable
- un MCA que obtiene un mensaje de una cola y lo descarta
- un manejador de mensajes no entregados colocando un mensaje de nuevo en una cola
- el servidor de mandatos que procesa una solicitud PCF-un intermediario que procesa una solicitud de publicación

- una aplicación de usuario que obtiene un mensaje de una cola: una aplicación de usuario que examina un mensaje en una cola

Cualquier aplicación, incluido el gestor de colas, puede añadir algunos de los datos de mensaje al informe de actividad después de la cabecera del informe. La cantidad de datos que se deben proporcionar si se envían algunos no es fija y la decide la aplicación. La información devuelta debe ser útil para la aplicación que procesa el informe de actividad. Los informes de actividad del gestor de colas devolverán con ellos las estructuras de cabecera IBM MQ estándar (empezando por 'MQH') contenidas en el mensaje original. Esto incluye, por ejemplo, cualquier cabecera MQRFH2 que se haya incluido en el mensaje original. Además, el gestor de colas devolverá una cabecera MQCFH encontrada, pero no los parámetros PCF asociados a ella. Esto da a las aplicaciones de supervisión una idea de lo que era el mensaje.

IBM i MQDXP (parámetro de salida de conversión de datos) en IBM i

Bloque de parámetros de salida de conversión de datos.

Visión general

Finalidad: La estructura MQDXP es un parámetro que el gestor de colas pasa a la salida de conversión de datos cuando se invoca la salida para convertir los datos del mensaje como parte del proceso de la llamada MQGET. Consulte la descripción de la llamada MQCONVX para obtener detalles de la salida de conversión de datos.

Conjunto de caracteres y codificación: los datos de caracteres de MQDXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId** . Los datos numéricos en MQDXP están en la codificación de máquina nativa; esto lo proporciona ENNAT.

Uso: solo la salida puede cambiar los campos *DXLEN*, *DXCC*, *DXREA* y *DXRES* en MQDXP; los cambios en otros campos se ignoran. Sin embargo, el campo *DXLEN* no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

Cuando el control vuelve al gestor de colas desde la salida, el gestor de colas comprueba los valores devueltos en MQDXP. Si los valores devueltos no son válidos, el gestor de colas continúa el proceso como si la salida hubiera devuelto XRFAIL en *DXRES* ; sin embargo, el gestor de colas ignora los valores de los campos *DXCC* y *DXREA* devueltos por la salida en este caso, y utiliza en su lugar los valores que esos campos tenían en *entrada* para la salida. Los siguientes valores en MQDXP hacen que se produzca este proceso:

- *DXRES* campo no XROK y no XRFAIL
- *DXCC* campo no CCOK y no CCWARN
- *DXLEN* campo menor que cero, o *DXLEN* campo cambiado cuando el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.
- “Campos” en la página 1480
- “Declaración RPG (copiar archivo CMQDXPH)” en la página 1484

Campos

La estructura MQDXP contiene los campos siguientes; los campos se describen en **orden alfabético**:

DXAOP (entero con signo de 10 dígitos)

Opciones de aplicación.

Se trata de una copia del campo *GMOPT* de la estructura MQGMO especificada por la aplicación que emite la llamada MQGET. Es posible que la salida tenga que examinarlos para determinar si se ha especificado la opción GMATM.

Es un campo de entrada para la salida.

DXCC (entero con signo de 10 dígitos)

Código de terminación.

Cuando se invoca la salida, contiene el código de terminación que se devolverá a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Siempre es CCWARN, porque el mensaje se ha truncado, o el mensaje requiere conversión y esto todavía no se ha realizado.

En la salida de la salida, este campo contiene el código de terminación que se devolverá a la aplicación en el parámetro **CMPCOD** de la llamada MQGET; sólo CCOK y CCWARN son válidos. Consulte la descripción del campo *DXREA* para obtener sugerencias sobre cómo la salida debe establecer este campo en la salida.

Es un campo de entrada/salida para la salida.

DXCSI (entero con signo de 10 dígitos)

Juego de caracteres necesario para la aplicación.

Este es el identificador de juego de caracteres codificado del juego de caracteres que necesita la aplicación que emite la llamada MQGET; consulte el campo *MDCSI* en la estructura MQMD para obtener más detalles. Si la aplicación especifica el valor especial CSQM en la llamada MQGET, el gestor de colas lo cambia por el identificador de juego de caracteres real del juego de caracteres utilizado por el gestor de colas, antes de invocar la salida.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *MDCSI* del descriptor de mensaje.

Es un campo de entrada para la salida.

DXENC (entero con signo de 10 dígitos)

Codificación numérica necesaria para la aplicación.

Esta es la codificación numérica que necesita la aplicación que emite la llamada MQGET; consulte el campo *MDENC* en la estructura MQMD para obtener más detalles.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *MDENC* del descriptor de mensaje.

Es un campo de entrada para la salida.

DXHCN (entero con signo de 10 dígitos)

Descriptor de conexión.

Se trata de un descriptor de conexión que se puede utilizar en la llamada MQXCNVC. Este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

DXLEN (entero con signo de 10 dígitos)

Longitud en bytes de datos de mensaje.

Cuando se invoca la salida, este campo contiene la longitud original de los datos del mensaje de aplicación. Si el mensaje se ha truncado para que quepa en el almacenamiento intermedio proporcionado por la aplicación, el tamaño del mensaje proporcionado a la salida será *menor* que el valor de *DXLEN*. El tamaño del mensaje proporcionado a la salida siempre lo proporciona el parámetro **INLEN** de la salida, independientemente de cualquier truncamiento que se haya podido producir.

El truncamiento se indica mediante el campo *DXREA* que tiene el valor RC2079 en la entrada de la salida.

La mayoría de las conversiones no tendrán que cambiar esta longitud, pero una salida puede hacerlo si es necesario; el valor establecido por la salida se devuelve a la aplicación en el parámetro **DATLEN** de la llamada MQGET. Sin embargo, esta longitud no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico. Esto se debe a que cambiar la longitud haría que los desplazamientos de segmentos posteriores en el mensaje lógico fueran incorrectos.

Tenga en cuenta que, si la salida desea cambiar la longitud de los datos, tenga en cuenta que el gestor de colas ya ha decidido si los datos del mensaje se ajustarán al almacenamiento intermedio de la aplicación, basándose en la longitud de los datos *no convertidos* . Esta decisión determina si el mensaje se elimina de la cola (o se mueve el cursor para examinar, para una solicitud de examinar) y

no se ve afectado por ningún cambio en la longitud de los datos causado por la conversión. Por este motivo, se recomienda que las salidas de conversión no provoquen un cambio en la longitud de los datos del mensaje de aplicación.

Si la conversión de caracteres implica un cambio de longitud, una serie se puede convertir en otra serie con la misma longitud en bytes, truncando los espacios en blanco finales o rellenando con espacios en blanco según sea necesario.

La salida no se invoca si el mensaje no contiene datos de mensaje de aplicación; por lo tanto, *DXLEN* siempre es mayor que cero.

Es un campo de entrada/salida para la salida.

DXREA (entero con signo de 10 dígitos)

Código de razón que califica *DXCC*.

Cuando se invoca la salida, contiene el código de razón que se devolverá a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Entre los valores posibles se encuentran RC2079, que indica que el mensaje se ha truncado para ajustarse al almacenamiento intermedio proporcionado por la aplicación, y RC2119, que indica que el mensaje requiere conversión pero que todavía no se ha realizado.

En la salida de la salida, este campo contiene la razón que debe devolverse a la aplicación en el parámetro **REASON** de la llamada MQGET; se recomienda lo siguiente:

- Si *DXREA* tenía el valor RC2079 en la entrada de la salida, los campos *DXREA* y *DXCC* no se deben modificar, independientemente de si la conversión se realiza correctamente o falla.

  (Si el campo *DXCC* no es CCOK, la aplicación que recupera el mensaje puede identificar una anomalía de conversión comparando los valores *MDENC* y *MDCSI* devueltos en el descriptor de mensaje con los valores solicitados; por el contrario, la aplicación no puede distinguir un mensaje truncado de un mensaje que acaba de ajustarse al almacenamiento intermedio. Por este motivo, RC2079 debe devolverse con preferencia a cualquiera de las razones que indican un error de conversión.)
- Si *DXREA* tenía cualquier otro valor en la entrada para la salida:
  - Si la conversión se realiza correctamente, *DXCC* debe establecerse en CCOK y *DXREA* debe establecerse en RCNONE.
  - Si la conversión falla, o el mensaje se expande y se tiene que truncar para que quepa en el almacenamiento intermedio, *DXCC* debe establecerse en CCWARN (o dejarse sin modificar), y *DXREA* establecerse en uno de los valores i de la lista siguiente, para indicar la naturaleza del error.

    Tenga en cuenta que, si el mensaje después de la conversión es demasiado grande para el almacenamiento intermedio, sólo se debe truncar si la aplicación que ha emitido la llamada MQGET ha especificado la opción GMATM:

    - Si ha especificado esta opción, se debe devolver la razón RC2079 .
    - Si no ha especificado esta opción, el mensaje debe devolverse sin convertir, con el código de razón RC2120.

Los códigos de razón de la lista siguiente se recomiendan para que los utilice la salida para indicar la razón por la que ha fallado la conversión, pero la salida puede devolver otros valores del conjunto de códigos RC* si se considera apropiado. Además, el rango de valores RC0900 a RC0999 se asignan para que los utilice la salida para indicar las condiciones en las que la salida desea comunicarse con la aplicación que emite la llamada MQGET.

Nota: Si el mensaje no se puede convertir correctamente, la salida debe devolver XRFAIL en el campo *DXRES* , para que el gestor de colas devuelva el mensaje no convertido. Esto es cierto independientemente del código de razón devuelto en el campo *DXREA* .

RC0900
(900, X'384 ') Valor más bajo para el código de razón definido por la aplicación.

RC0999
(999, X'3E7') Valor más alto para el código de razón definido por la aplicación.

RC2120
(2120, X'848 ') Datos convertidos demasiado grandes para almacenamiento intermedio.

RC2119
(2119, X'847') Los datos del mensaje no se han convertido.

RC2111
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2113
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

RC2114
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

RC2112
(2112, X'840') Codificación de entero de origen no reconocida.

RC2115
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2117
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

RC2118
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

RC2116
(2116, X'844') Codificación de entero de destino no reconocida.

RC2079
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

Es un campo de entrada/salida para la salida.

DXRES (entero con signo de 10 dígitos)

Respuesta de la salida.

Esto lo establece la salida para indicar el éxito o no de la conversión. Debe ser uno de los siguientes:

XRC
La conversión ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *DXCC* en la salida de la salida
- El valor del campo *DXREA* en la salida de la salida
- El valor del campo *DXLEN* en la salida de la salida
- El contenido del almacenamiento intermedio de salida de la salida *OUTBUF*. El número de bytes devueltos es el menor del parámetro **OUTLEN** de la salida y el valor del campo *DXLEN* en la salida de la salida

Si los campos *MDENC* y *MDCSI* del parámetro de descriptor de mensaje de la salida *ambos* no se modifican, el gestor de colas devuelve:

- El valor de los campos *MDENC* y *MDCSI* en la estructura MQDXP en *entrada* a la salida

Si se ha cambiado uno o ambos campos *MDENC* y *MDCSI* en el parámetro de descriptor de mensaje de la salida, el gestor de colas devuelve:

- El valor de los campos *MDENC* y *MDCSI* en el parámetro de descriptor de mensaje de la salida en la salida de la salida
-

XRFAIL
La conversión no ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *DXCC* en la salida de la salida
- El valor del campo *DXREA* en la salida de la salida
- El valor del campo *DXLEN* en *entrada* a la salida
- El contenido del almacenamiento intermedio de entrada de la salida *INBUF*. El número de bytes devueltos lo proporciona el parámetro **INLEN**

Si la salida ha modificado *INBUF*, los resultados no están definidos.

DXRES es un campo de salida de la salida.

DXSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DXSIDV
Identificador de la estructura de parámetros de salida de conversión de datos.

Es un campo de entrada para la salida.

DXVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DXVER1
Número de versión para la estructura de parámetros de salida de conversión de datos.

La constante siguiente especifica el número de versión de la versión actual:

DXVERC
Versión actual de la estructura de parámetros de salida de conversión de datos.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el campo *DXVER* es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

DXXOP (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor es `0`.

Declaración RPG (copiar archivo 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 (Convertir caracteres) en IBM i

La llamada MQXCNVC convierte los caracteres de un juego de caracteres a otro.

Esta llamada forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ . Nota: Esta llamada sólo se puede utilizar desde una salida de conversión de datos.

- "Sintaxis" en la página 1485
- "Parámetros" en la página 1485
- "Invocación de RPG (ILE)" en la página 1489

Sintaxis

```
MQXCNVC HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSI, TGTLEN,
   TGTBUF, DATLEN, CMPCOD, REASON)
```

Parámetros

La llamada MQXCNVC tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. Normalmente debe ser el descriptor de contexto pasado a la salida de conversión de datos en el campo DXHCN de la estructura MQDXP; este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

En IBM i, se puede especificar el siguiente valor especial para HCONN:

HCDEFH
Manejador de conexión predeterminado.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQXCNVC.

Se pueden especificar cero o más de las opciones descritas más adelante en esta sección. Si se necesita más de uno, se pueden añadir los valores (no añadir la misma constante más de una vez).

Opción de conversión predeterminada: la siguiente opción controla el uso de la conversión de caracteres predeterminada:

DCCDEF
Conversión predeterminada.

Esta opción especifica que se puede utilizar la conversión de caracteres por omisión si uno o ambos de los juegos de caracteres especificados en la llamada no están soportados. Esto permite al gestor de colas utilizar un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de caracteres especificado, al convertir la serie.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir la serie es que algunos caracteres pueden convertirse incorrectamente. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres especificado como al juego de caracteres predeterminado.

Los juegos de caracteres predeterminados se definen mediante una opción de configuración cuando se instala o se reinicia el gestor de colas.

Si no se especifica DCCDEF, el gestor de colas sólo utiliza los juegos de caracteres especificados para convertir la serie, y la llamada falla si uno o ambos juegos de caracteres no están soportados.

Opción de relleno: la opción siguiente permite al gestor de colas rellenar la serie convertida con espacios en blanco o descartar caracteres finales insignificantes, para que la serie convertida se ajuste al almacenamiento intermedio de destino:

DCCFIL

Rellene el almacenamiento intermedio de destino.

Esta opción solicita que la conversión tenga lugar de forma que el almacenamiento intermedio de destino se llene completamente:

- Si la serie se contrae cuando se convierte, se añaden espacios en blanco finales para rellenar el almacenamiento intermedio de destino.
- Si la serie se expande cuando se convierte, los caracteres finales que no son significativos se descartan para que la serie convertida se ajuste al almacenamiento intermedio de destino. Si esto se puede realizar correctamente, la llamada se completa con CCOK y el código de razón RCNONE.

  Si hay muy pocos caracteres finales insignificantes, la mayor parte de la serie que quepa se coloca en el almacenamiento intermedio de destino y la llamada se completa con CCWARN y el código de razón RC2120.

  Los caracteres insignificantes son:

  – Blancos de cola
  – Caracteres que siguen al primer carácter nulo de la serie (pero excluyendo el propio primer carácter nulo)
- Si la serie, `TGTCSI` y `TGTLEN` son tales que el almacenamiento intermedio de destino no se puede establecer completamente con caracteres válidos, la llamada falla con CCFAIL y el código de razón RC2144. Esto puede ocurrir cuando `TGTCSI` es un juego de caracteres DBCS puro (como UTF-16), pero `TGTLEN` especifica una longitud que es un número impar de bytes.
- `TGTLEN` puede ser menor o mayor que `SRCLEN`. Al volver de MQXCNVC, `DATLEN` tiene el mismo valor que `TGTLEN`.

Si no se especifica esta opción:

- La serie puede contraerse o expandirse dentro del almacenamiento intermedio de destino según sea necesario. Los caracteres finales insignificantes no se añaden ni descartan.

  Si la serie convertida se ajusta al almacenamiento intermedio de destino, la llamada se completa con CCOK y el código de razón RCNONE.

  Si la serie convertida es demasiado grande para el almacenamiento intermedio de destino, la mayor parte de la serie que quepa se coloca en el almacenamiento intermedio de destino y la llamada se completa con CCWARN y el código de razón RC2120. Tenga en cuenta que en este caso se pueden devolver menos de *TGTLEN* bytes.
- `TGTLEN` puede ser menor o mayor que `SRCLEN`. Al volver de MQXCNVC, `DATLEN` es menor o igual que `TGTLEN`.

Opciones de codificación: se pueden utilizar las opciones siguientes para especificar las codificaciones de enteros de las series de origen y destino. La codificación relevante sólo se utiliza cuando el identificador de juego de caracteres correspondiente indica que la representación del juego de caracteres en el almacenamiento principal depende de la codificación utilizada para los enteros

binarios. Esto sólo afecta a determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16).

La codificación se ignora si el juego de caracteres es un juego de caracteres de un solo byte (SBCS), o un juego de caracteres de varios bytes con representación en el almacenamiento principal que no depende de la codificación de enteros.

Sólo debe especificarse uno de los valores DCCS*, combinado con uno de los valores DCCT*:

DCCSNA
La codificación de origen es el valor predeterminado para el entorno y el lenguaje de programación.

DCCSNO
La codificación de origen es normal.

DCCSRE
La codificación de origen se invierte.

DCCSUN
La codificación de origen no está definida.

DCCTNA
La codificación de destino es el valor predeterminado para el entorno y el lenguaje de programación.

DCCTNO
La codificación de destino es normal.

DCCTRE
La codificación de destino se invierte.

DCCTUN
La codificación de destino no está definida.

Los valores de codificación definidos anteriormente se pueden añadir directamente al campo OPTS . Sin embargo, si la codificación de origen o destino se obtiene del campo MDENC en el MQMD u otra estructura, se debe realizar el proceso siguiente:

1. La codificación de enteros debe extraerse del campo MDENC eliminando las codificaciones decimal flotante y empaquetado; consulte “Análisis de codificaciones en IBM i” en la página 1469 para obtener detalles sobre cómo hacerlo.
2. La codificación de enteros resultante del paso 1 debe multiplicarse por el factor adecuado antes de añadirse al campo OPTS . Estos factores son:

   DCCSFA
   Factor para codificación de origen

   DCCTFA
   Factor para codificación de destino

Si no se especifica, las opciones de codificación toman como valor predeterminado sin definir (DCC* UN). En la mayoría de los casos, esto no afecta a la finalización satisfactoria de la llamada MQXCNVC. Sin embargo, si el juego de caracteres correspondiente es un juego de caracteres multibyte con representación que depende de la codificación (por ejemplo, un juego de caracteres UTF-16), la llamada falla con el código de razón RC2112 o RC2116 según corresponda.

Opción predeterminada: si no se especifica ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

DCCNON
No se ha especificado ninguna opción.

DCCNON está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

SRCCSI (entero con signo de 10 dígitos)-entrada

Identificador de juego de caracteres codificado de serie antes de la conversión.

Es el identificador de juego de caracteres codificado de la serie de entrada en SRCBUF.

SRCLEN (entero con signo de 10 dígitos)-entrada

Longitud de la serie antes de la conversión.

Es la longitud en bytes de la serie de entrada en SRCBUF ; debe ser cero o mayor.

SRCBUF (serie de caracteres de 1 byte x SRCLEN)-entrada

Serie que se va a convertir.

Es el almacenamiento intermedio que contiene la serie que se va a convertir de un juego de caracteres a otro.

TGTCSI (entero con signo de 10 dígitos)-entrada

Identificador de juego de caracteres codificado de la serie después de la conversión.

Es el identificador de juego de caracteres codificado del juego de caracteres al que se va a convertir SRCBUF .

TGTLEN (entero con signo de 10 dígitos)-entrada

Longitud del almacenamiento intermedio de salida.

Es la longitud en bytes del almacenamiento intermedio de salida TGTBUF ; debe ser cero o mayor. Puede ser menor o mayor que SRCLEN.

TGTBUF (serie de caracteres de 1 byte x TGTLEN)-salida

Serie después de la conversión.

Esta es la serie después de que se haya convertido al juego de caracteres definido por TGTCSI. La serie convertida puede ser más corta o más larga que la serie no convertida. El parámetro **DATLEN** indica el número de bytes válidos devueltos.

DATLEN (entero con signo de 10 dígitos)-salida

Longitud de la serie de salida.

Es la longitud de la serie devuelta en el almacenamiento intermedio de salida TGTBUF. La serie convertida puede ser más corta o más larga que la serie no convertida.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK
Realización satisfactoria.

CCWARN
Aviso (finalización parcial).

CCFAIL
La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Si CMPCOD es CCOK:

RCNONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCWARN:

RC2120
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

Si CMPCOD es CCFAIL:

RC2010
(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2150
(2150, X'866') La serie DBCS no es válida.

RC2018
(2018, X'7E2') El manejador de conexión no es válido.

RC2046
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2102
(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2145
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

RC2111
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2112
(2112, X'840') Codificación de entero de origen no reconocida.

RC2143
(2143, X'85F') Parámetro de longitud de origen no válido.

RC2071
(2071, X'817') No hay suficiente almacenamiento disponible.

RC2146
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

RC2115
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2116
(2116, X'844') Codificación de entero de destino no reconocida.

RC2144
(2144, X'860 ') El parámetro de longitud de destino no es válido.

RC2195
(2195, X'893') Se ha producido un error inesperado.

Para obtener más información sobre estos códigos de razón, consulte “Códigos de retorno para IBM i (ILE RPG)” en la página 1462.

Invocación de RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQXCNVC(HCONN : OPTS : SRCCSI :
C                                       SRCLEN : SRCBUF : TGTCSI :
C                                       TGTLEN : TGTBUF : DATLEN :
C                                       CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
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 (Salida de conversión de datos) en IBM i

Esta definición de llamada describe los parámetros que se pasan a la salida de conversión de datos.

El gestor de colas no proporciona ningún punto de entrada denominado MQCONVX (consulte la nota de uso "11" en la página 1492).

Esta definición forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ .

- "Sintaxis" en la página 1490
- "Notas de uso" en la página 1490
- "Parámetros" en la página 1492
- "Invocación de RPG (ILE)" en la página 1493

Sintaxis

MQCONVX ***(MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF)***

Notas de uso

1. Una salida de conversión de datos es una salida escrita por el usuario que recibe el control durante el proceso de una llamada MQGET. La función realizada por la salida de conversión de datos la define el proveedor de la salida; sin embargo, la salida debe ajustarse a las reglas descritas aquí y en la estructura de parámetros asociada MQDXP.

   El entorno determina los lenguajes de programación que se pueden utilizar para una salida de conversión de datos.
2. La salida sólo se invoca si se cumplen *todas* las sentencias siguientes:
   - La opción GMCONV se especifica en la llamada MQGET
   - El campo *MDFMT* del descriptor de mensaje no es FMNONE
   - El mensaje no está ya en la representación necesaria; es decir, uno o ambos *MDCSI* y *MDENC* del mensaje son diferentes del valor especificado por la aplicación en el descriptor de mensaje proporcionado en la llamada MQGET
   - El gestor de colas todavía no ha realizado la conversión correctamente
   - La longitud del almacenamiento intermedio de la aplicación es mayor que cero
   - La longitud de los datos del mensaje es mayor que cero
   - El código de razón hasta ahora durante la operación MQGET es RCNONE o RC2079
3. Cuando se está escribiendo una salida, se debe tener en cuenta la codificación de la salida de una forma que le permita convertir los mensajes que se han truncado. Los mensajes truncados pueden aparecer de las siguientes maneras:
   - La aplicación receptora proporciona un almacenamiento intermedio que es menor que el mensaje, pero especifica la opción GMATM en la llamada MQGET.

     En este caso, el campo *DXREA* del parámetro **MQDXP** en la entrada de la salida tendrá el valor RC2079.

- El remitente del mensaje lo ha truncado antes de enviarlo. Esto puede suceder con los mensajes de informe, por ejemplo (consulte “Conversión de mensajes de informe en IBM i” en la página 1479 para obtener más detalles).

  En este caso, el campo *DXREA* del parámetro **MQDXP** en la entrada de la salida tendrá el valor RCNONE (si la aplicación receptora ha proporcionado un almacenamiento intermedio lo suficientemente grande para el mensaje).

Por lo tanto, el valor del campo *DXREA* en la entrada a la salida no siempre se puede utilizar para decidir si el mensaje se ha truncado.

La característica distintiva de un mensaje truncado es que la longitud proporcionada a la salida en el parámetro **INLEN** será *menor que* la longitud implícita por el nombre de formato contenido en el campo *MDFMT* del descriptor de mensaje. Por lo tanto, la salida debe comprobar el valor de *INLEN* antes de intentar convertir cualquiera de los datos; la salida *no* debe suponer que se ha proporcionado la cantidad completa de datos implícita en el nombre de formato.

Si la salida no se ha grabado para convertir mensajes truncados, y **INLEN** es menor que el valor esperado, la salida debe devolver XRFAIL en el campo *DXRES* del parámetro **MQDXP** , con el campo *DXCC* establecido en CCWARN y el campo *DXREA* establecido en RC2110.

Si la salida *ha* se ha grabado para convertir mensajes truncados, la salida debe convertir la mayor cantidad posible de datos (consulte la siguiente nota de uso), teniendo cuidado de no intentar examinar o convertir datos más allá del final de *INBUF*. Si la conversión se completa correctamente, la salida debe dejar el campo *DXREA* en el parámetro **MQDXP** sin modificar. Esto devuelve RC2079 si el gestor de colas del receptor ha truncado el mensaje y RCNONE si el emisor del mensaje ha truncado el mensaje.

También es posible que un mensaje expanda *durante la conversión de* , hasta el punto en el que es mayor que *OUTBUF*. En este caso, la salida debe decidir si trunca el mensaje; el campo *DXAOP* del parámetro **MQDXP** indicará si la aplicación receptora ha especificado la opción GMATM.

4. Por lo general, se recomienda que se conviertan todos los datos del mensaje proporcionado a la salida en *INBUF* , o que no se convierta ninguno de ellos. Sin embargo, se produce una excepción a esto si el mensaje se trunca, ya sea antes de la conversión o durante la conversión; en este caso puede haber un elemento incompleto al final del almacenamiento intermedio (por ejemplo: un byte de un carácter de doble byte o 3 bytes de un entero de 4 bytes). En esta situación, se recomienda que se omita el elemento incompleto y que los bytes no utilizados en *OUTBUF* se establezcan en nulos. Sin embargo, los elementos o caracteres completos dentro de una matriz o serie *deben* convertirse.
5. Cuando se necesita una salida por primera vez, el gestor de colas intenta cargar un objeto que tiene el mismo nombre que el formato (aparte de las extensiones). El objeto cargado debe contener la salida que procesa los mensajes con ese nombre de formato. Se recomienda que el nombre de salida y el nombre del objeto que contiene la salida sean idénticos, aunque no todos los entornos lo requieran.
6. Se carga una nueva copia de la salida cuando una aplicación intenta recuperar el primer mensaje que utiliza ese *MDFMT* desde que la aplicación se conectó al gestor de colas. Una nueva copia también se puede cargar en otras ocasiones, si el gestor de colas ha descartado una copia cargada anteriormente. Por esta razón, una salida no debe intentar utilizar el almacenamiento estático para comunicar información de una invocación de la salida a la siguiente-la salida se puede descargar entre las dos invocaciones.
7. Si hay una salida proporcionada por el usuario con el mismo nombre que uno de los formatos incorporados soportados por el gestor de colas, la salida proporcionada por el usuario no sustituye la rutina de conversión incorporada. Las únicas circunstancias en las que se invoca una salida de este tipo son:
   - Si la rutina de conversión incorporada no puede manejar conversiones a o desde los *MDCSI* o *MDENC* implicados, o
   - Si la rutina de conversión incorporada no ha podido convertir los datos (por ejemplo, porque hay un campo o carácter que no se puede convertir).

8. El ámbito de la salida depende del entorno. Los nombres de *MDFMT* deben elegirse para minimizar el riesgo de conflictos con otros formatos. Se recomienda que empiecen por caracteres que identifiquen la aplicación que define el nombre de formato.
9. La salida de conversión de datos se ejecuta en un entorno como el del programa que ha emitido la llamada MQGET; el entorno incluye el espacio de direcciones y el perfil de usuario (si procede). El programa podría ser un agente de canal de mensajes que envía mensajes a un gestor de colas de destino que no soporta la conversión de mensajes. La salida no puede comprometer la integridad del gestor de colas, ya que no se ejecuta en el entorno del gestor de colas.
10. La única llamada MQI que puede utilizar la salida es MQXCNVC; el intento de utilizar otras llamadas MQI falla con el código de razón RC2219u otros errores imprevisibles.
11. El gestor de colas no proporciona ningún punto de entrada denominado MQCONVX. El nombre de la salida debe ser el mismo que el nombre de formato (el nombre contenido en el campo *MDFMT* en MQMD), aunque esto no es necesario en todos los entornos.

Parámetros

La llamada MQCONVX tiene los parámetros siguientes:

MQDXP (MQDXP)-entrada/salida

Bloque de parámetros de salida de conversión de datos.

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar el resultado de la conversión. Consulte "MQDXP (parámetro de salida de conversión de datos) en IBM i" en la página 1480 para obtener detalles de los campos de esta estructura.

MQMD (MQMD)-entrada/salida

Descriptor de mensaje.

En la entrada a la salida, este es el descriptor de mensaje que se devolvería a la aplicación si no se realizara ninguna conversión. Por lo tanto, contiene *MDFMT*, *MDENC*y *MDCSI* del mensaje no convertido contenido en *INBUF*.

Nota: El parámetro **MQMD** pasado a la salida es siempre la versión más reciente de MQMD soportada por el gestor de colas que invoca la salida. Si la salida está pensada para ser portable entre distintos entornos, la salida debe comprobar el campo *MDVER* en *MQMD* para verificar que los campos a los que la salida necesita acceder están presentes en la estructura.

En IBM i, a la salida se le pasa un MQMD de version-2 .

En la salida, la salida debe cambiar los campos *MDENC* y *MDCSI* por los valores solicitados por la aplicación, si la conversión ha sido satisfactoria; estos cambios se reflejarán de nuevo en la aplicación. Cualquier otro cambio que realice la salida en la estructura se ignorará; no se reflejarán de nuevo en la aplicación.

Si la salida devuelve XROK en el campo *DXRES* de la estructura MQDXP, pero no cambia los campos *MDENC* o *MDCSI* del descriptor de mensaje, el gestor de colas devuelve para esos campos los valores que los campos correspondientes de la estructura MQDXP tenían en la entrada de la salida.

INLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes de *INBUF*.

Es la longitud del almacenamiento intermedio de entrada *INBUF*y especifica el número de bytes que debe procesar la salida. *INLEN* es la longitud menor de los datos del mensaje antes de la conversión y la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

INBUF (serie de bits de 1 byte x INLEN)-entrada

Almacenamiento intermedio que contiene el mensaje no convertido.

Contiene los datos del mensaje antes de la conversión. Si la salida no puede convertir los datos, el gestor de colas devuelve el contenido de este almacenamiento intermedio a la aplicación una vez completada la salida.

Nota: La salida no debe alterar *INBUF* ; si se altera este parámetro, los resultados no están definidos.

OUTLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes de *OUTBUF*.

Es la longitud del almacenamiento intermedio de salida *OUTBUF*y es la misma que la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

OUTBUF (serie de bits de 1 byte x OUTLEN)-salida

Almacenamiento intermedio que contiene el mensaje convertido.

En la salida de la salida, si la conversión ha sido satisfactoria (tal como indica el valor XROK en el campo *DXRES* del parámetro **MQDXP**), **OUTBUF** contiene los datos de mensaje que se van a entregar a la aplicación, en la representación solicitada. Si la conversión no ha sido satisfactoria, se ignoran los cambios que la salida ha realizado en este almacenamiento intermedio.

Invocación de RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     exitname(MQDXP : MQMD : INLEN :
C                                        INBUF : OUTLEN : OUTBUF)
```

La definición de prototipo para la llamada es:

```
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
```

Fin de la interfaz de programación sensible al producto

Referencia de salidas de usuarios, salidas de API y servicios instalables

Utilice la información de esta sección para ayudarle a desarrollar las salidas de usuario, las salidas de API y las aplicaciones de servicios instalables:

- “Estructura MQIEP” en la página 1494
- “Referencia de salida de conversión de datos” en la página 1497
- “MQ_PUBLISH_EXIT - Salida de publicación” en la página 1501
- “Llamadas de salida de canal y estructuras de datos” en la página 1509
- “Referencia a la salida de la API” en la página 1602
- “Información de consulta sobre la interfaz de servicios instalables” en la página 1664

Conceptos relacionados

Salidas de usuario, salidas de API y servicios instalables de IBM MQ

Tareas relacionadas

Extensión de recursos del gestor de colas

Estructura MQIEP

La estructura MQIEP contiene un punto de entrada para cada llamada de función que se permite realizar a las salidas.

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQIEP_STRUC_ID

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQIEP_VERSION_1
Número de versión de estructura de la versión 1.

MQIEP_CURRENT_VERSION
Versión actual de la estructura.

StrucLength

Tipo: MQLONG

Tamaño de la estructura MQIEP en bytes. El valor es el siguiente:

MQIEP_LENGTH_1

Distintivos

Tipo: MQLONG

Proporciona información sobre las direcciones de función. Un distintivo para indicar si la biblioteca tiene hebras puede utilizarse con un distintivo para indicar si la biblioteca es una biblioteca de cliente o de servidor.

El valor siguiente se utiliza para no especificar información de biblioteca:

MQIEPF_NONE

Se utiliza uno de los valores siguientes para especificar si la biblioteca compartida es con hebras o no con hebras:

MQIEPF_NON_THREADED_LIBRARY
Una biblioteca compartida sin hebras

MQIEPF_THREADED_LIBRARY
Una biblioteca compartida con hebras

Se utiliza uno de los valores siguientes para especificar si la biblioteca compartida es un cliente o una biblioteca compartida de servidor:

MQIEPF_BIBLIOTECA_CLIENTE
Una biblioteca compartida de cliente

MQIEPF_BIBLIOTECA_LOCAL
Una biblioteca compartida de servidor

Reserved

Tipo: MQPTR

Llamada MQBACK_Call
Tipo: PMQ_BACK_CALL

Dirección de la llamada MQBACK.

MQBEGIN_Llamada
Tipo: PMQ_BEGIN_CALL

Dirección de la llamada MQBEGIN.

MQBUFMH_Llamada
Tipo: PMQ_BUFMH_CALL

Dirección de la llamada MQBUFMH.

MQCB_Llamada
Tipo: PMQ_CB_CALL

Dirección de la llamada MQCB.

Llamada MQCLOSE_Call
Tipo: PMQ_CLOSE_CALL

Dirección de la llamada MQCLOSE.

MQCMIT_Llamada
Tipo: PMQ_CMIT_CALL

Dirección de la llamada MQCMIT.

MQCONN_Llamada
Tipo: PMQ_CONN_CALL

Dirección de la llamada MQCONN.

MQCONNX_Llamada
Tipo: PMQ_CONNX_CALL

Dirección de la llamada MQCONNX.

MQCRTMH_Llamada
Tipo: PMQ_CRTMH_CALL

Dirección de la llamada MQCRTMH.

MQCTL_Llamada
Tipo: PMQ_CTL_CALL

Dirección de la llamada MQCTL.

MQDISC_Llamada
Tipo: PMQ_DISC_CALL

Dirección de la llamada MQDISC.

Llamada MQDLTMH_Call
Tipo: PMQ_DLTMH_CALL

Dirección de la llamada MQDLTMH.

Llamada MQDLTMP_Call
Tipo: PMQ_DLTMP_CALL

Dirección de la llamada MQDLTMP.

Llamada MQGET_Call
Tipo: PMQ_GET_CALL

Dirección de la llamada MQGET.

MQINQ_Llamada
Tipo: PMQ_INQ_CALL

Dirección de la llamada MQINQ.

MQINQMP_Llamada
Tipo: PMQ_INQMP_CALL

Dirección de la llamada MQINQMP.

Llamada MQMHBUF_Call
Tipo: PMQ_MHBUF_CALL

Dirección de la llamada MQMHBUF.

MQOPEN_Llamada
Tipo: PMQ_OPEN_CALL

Dirección de la llamada MQOPEN.

Llamada MQPUT_Call
Tipo: PMQ_PUT_CALL

Dirección de la llamada MQPUT.

MQPUT1_Call
Tipo: PMQ_PUT1_CALL

Dirección de la llamada MQPUT1 .

MQSET_Call
Tipo: PMQ_SET_CALL

Dirección de la llamada MQSET.

Llamada MQSETMP_Call
Tipo: PMQ_SETMP_CALL

Dirección de la llamada MQSETMP.

MQSTAT_Llamada
Tipo: PMQ_STAT_CALL

Dirección de la llamada MQSTAT.

MQSUB_Llamada
Tipo: PMQ_SUB_CALL

Dirección de la llamada MQSUB.

MQSUBRQ_Llamada
Tipo: PMQ_SUBRQ_CALL

Dirección de la llamada MQSUBRQ.

MQXCNVC_Llamada
Tipo: PMQ_XCNVC_CALL

Dirección de la llamada MQXCNVC.

MQXCLWLN_Llamada
Tipo: PMQ_XCLWLN_CALL

Dirección de la llamada MQXCLWLN.

MQXDX_Llamada
Tipo: PMQ_XDX_CALL

Dirección de la llamada MQXDX.

MQXEP_Llamada
Tipo: PMQ_XEP_CALL

Dirección de la llamada MQXEP.

MQZEP_Llamada

Tipo: PMQ_ZEP_CALL

Dirección de la llamada MQZEP.

Declaración C

```
struct tagMQIEP {
   MQCHAR4          StrucId;          /* Structure identifier */
   MQLONG           Version;          /* Structure version number */
   MQLONG           StrucLength;      /* Structure length */
   MQLONG           Flags;            /* Flags */
   MQPTR            Reserved;         /* Reserved */
   PMQ_BACK_CALL    MQBACK_Call;      /* Address of MQBACK */
   PMQ_BEGIN_CALL   MQBEGIN_Call;     /* Address of MQBEGIN */
   PMQ_BUFMH_CALL   MQBUFMH_Call;     /* Address of MQBUFMH */
   PMQ_CB_CALL      MQCB_Call;        /* Address of MQCB */
   PMQ_CLOSE_CALL   MQCLOSE_Call;     /* Address of MQCLOSE */
   PMQ_CMIT_CALL    MQCMIT_Call;      /* Address of MQCMIT */
   PMQ_CONN_CALL    MQCONN_Call;      /* Address of MQCONN */
   PMQ_CONNX_CALL   MQCONNX_Call;     /* Address of MQCONNX */
   PMQ_CRTMH_CALL   MQCRTMH_Call;     /* Address of MQCRTMH */
   PMQ_CTL_CALL     MQCTL_Call;       /* Address of MQCTL */
   PMQ_DISC_CALL    MQDISC_Call;      /* Address of MQDISC */
   PMQ_DLTMH_CALL   MQDLTMH_Call;     /* Address of MQDLTMH */
   PMQ_DLTMP_CALL   MQDLTMP_Call;     /* Address of MQDLTMP */
   PMQ_GET_CALL     MQGET_Call;       /* Address of MQGET */
   PMQ_INQ_CALL     MQINQ_Call;       /* Address of MQINQ */
   PMQ_INQMP_CALL   MQINQMP_Call;     /* Address of MQINQMP */
   PMQ_MHBUF_CALL   MQMHBUF_Call;     /* Address of MQMHBUF */
   PMQ_OPEN_CALL    MQOPEN_Call;      /* Address of MQOPEN */
   PMQ_PUT_CALL     MQPUT_Call;       /* Address of MQPUT */
   PMQ_PUT1_CALL    MQPUT1_Call;      /* Address of MQPUT1 */
   PMQ_SET_CALL     MQSET_Call;       /* Address of MQSET */
   PMQ_SETMP_CALL   MQSETMP_Call;     /* Address of MQSETMP */
   PMQ_STAT_CALL    MQSTAT_Call;      /* Address of MQSTAT */
   PMQ_SUB_CALL     MQSUB_Call;       /* Address of MQSUB */
   PMQ_SUBRQ_CALL   MQSUBRQ_Call;     /* Address of MQSUBRQ */
   PMQ_XCLWLN_CALL  MQXCLWLN_Call;    /* Address of MQXCLWLN */
   PMQ_XCNVC_CALL   MQXCNVC_Call;     /* Address of MQXCNVC */
   PMQ_XDX_CALL     MQXDX_Call;       /* Address of MQXDX */
   PMQ_XEP_CALL     MQXEP_Call;       /* Address of MQXEP */
   PMQ_ZEP_CALL     MQZEP_Call;       /* Address of MQZEP */
};
```

Referencia de salida de conversión de datos

Para z/OS, debe escribir salidas de conversión de datos en lenguaje ensamblador. Para otras plataformas, se recomienda utilizar el lenguaje de programación C.

Para ayudarle a crear un programa de salida de conversión de datos, se proporcionan los recursos siguientes:

- Un archivo de origen de esqueleto
- Una llamada de conversión de caracteres
- Un programa de utilidad que crea un fragmento de código que realiza la conversión de datos en estructuras de tipo de datos. Este programa de utilidad sólo toma la entrada C. En z/OS, genera código de ensamblador.

Para el procedimiento para escribir los programas, consulte:

- IBM i Escritura de un programa de salida de conversión de datos para IBM i
- z/OS Escritura de un programa de salida de conversión de datos para IBM MQ for z/OS
- Escritura de una salida de conversión de datos para IBM MQ en sistemas UNIX and Linux
- Escritura de una salida de conversión de datos para IBM MQ for Windows

Archivo de origen de esqueleto

Estos se pueden utilizar como punto de partida al escribir un programa de salida de conversión de datos.

Los archivos suministrados se listan en Tabla 816 en la página 1498.

Tabla 816. Archivos de origen de esqueleto

Plataforma	Archivo
AIX	amqsvfc0.c
IBM i	QMQMSAMP/QCSRC (AMQSVFC4)
Linux	amqsvfc0.c
Solaris	amqsvfc0.c
Sistemas Windows	amqsvfc0.c
z/OS	CSQ4BAX8 (“1” en la página 1498) CSQ4BAX9 (“2” en la página 1498) CSQ4CAX9 (“3” en la página 1498)

Notas:

1. Ilustra la llamada MQXCVNC.
2. Un derivador para los fragmentos de código generados por el programa de utilidad para su uso en todos los entornos excepto CICS.
3. Un derivador para los fragmentos de código generados por el programa de utilidad para su uso en el entorno CICS .

Llamada de conversión de caracteres

Utilice la llamada MQXCNVC (convertir caracteres) desde dentro de un programa de salida de conversión de datos para convertir datos de mensajes de caracteres de un juego de caracteres a otro. Para determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16), se deben utilizar las opciones adecuadas.

No se pueden realizar otras llamadas MQI desde dentro de la salida; un intento de realizar una llamada de este tipo falla con el código de razón MQRC_CALL_IN_PROGRESS.

Consulte “MQXCNVC-Convertir caracteres” en la página 935 para obtener más información sobre la llamada MQXCNVC y las opciones adecuadas.

Utilidad para crear código de salida de conversión

Utilice esta información para obtener más información sobre cómo crear código de salida de conversión.

Los mandatos para crear el código de salida de conversión son:

IBM i
CVTMQMDTA (Convertir tipo de datos IBM MQ)

Sistemas Windows, UNIX and Linux
`crtmqcvx` (Crear IBM MQ conversion-exit)

z/OS
CSQUCVX

El mandato para la plataforma produce un fragmento de código que realiza la conversión de datos en estructuras de tipo de datos, para su uso en el programa de salida de conversión de datos. El mandato toma un archivo que contiene una o más definiciones de estructura de lenguaje C. z/OS En z/OS, genera un conjunto de datos que contiene fragmentos de código de ensamblador y funciones de conversión. En otras plataformas, genera un archivo con una función C para convertir cada definición de estructura. En z/OS, el programa de utilidad requiere acceso a la biblioteca de tiempo de ejecución LE/370 SCEERUN.

Invocación del programa de utilidad CSQUCVX en z/OS

z/OS

La Figura 10 en la página 1499 muestra un ejemplo del JCL utilizado para invocar el programa de utilidad CSQUCVX.

```
//CVX    EXEC PGM=CSQUCVX
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//         DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP  DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT  DD DISP=OLD,DSN=MY.MQSERIES.EXITS(MSG1)
```

Figura 10. JCL de ejemplo utilizado para invocar el programa de utilidad CSQUCVX

z/OS sentencias de definición de datos

z/OS

El programa de utilidad CSQUCVX requiere sentencias DD con los siguientes nombres DD:

Tabla 817. Nombres y descripciones de sentencias de definición de datos

Sentencia DD	Descripción
SYSPRINT	Especifica un conjunto de datos o una clase de spool de impresión para informes y mensajes de error.
CSQUINP	Especifica el conjunto de datos particionados que contiene las definiciones de las estructuras de datos que se van a convertir.
CSQUOUT	Especifica el conjunto de datos particionados donde se van a grabar los fragmentos de código de conversión. La longitud de registro lógico (LRECL) debe ser 80 y el formato de registro (RECFM) debe ser FB.

Mensajes de error en sistemas Windows, UNIX and Linux

El mandato `crtmqcvx` devuelve mensajes en el rango de AMQ7953 a AMQ7970.

Estos mensajes se listan en Mensajes y códigos de razón *IBM MQ Mensajes*.

Hay dos tipos principales de error:

- Errores graves, como errores de sintaxis, cuando el proceso no puede continuar.

  Se visualiza un mensaje en la pantalla que indica el número de línea del error en el archivo de entrada. Es posible que el archivo de salida se haya creado parcialmente.
- Otros errores cuando se visualiza un mensaje que indica que se ha encontrado un problema pero que el análisis de la estructura puede continuar.

El archivo de salida se ha creado y contiene información de error sobre los problemas que se han producido. Esta información de error tiene el prefijo `#error` para que el código generado no sea aceptado por ningún compilador sin intervención para rectificar los problemas.

Sintaxis válida

El archivo de entrada para el programa de utilidad debe ajustarse a la sintaxis del lenguaje C.

Si no está familiarizado con C, consulte el ejemplo C de este tema.

Además, tenga en cuenta las reglas siguientes:

- typedef sólo se reconoce antes de la palabra clave struct.
- Se necesita un código de estructura en las declaraciones de estructura.
- Puede utilizar corchetes vacíos [] para indicar una serie o matriz de longitud variable al final de un mensaje.
- Las matrices multidimensionales y las matrices de series no están soportadas.
- Se reconocen los siguientes tipos de datos adicionales:
  - MQBOOL
  - MQBYTE
  - MQCHAR
  - MQFLOAT32
  - MQFLOAT64
  - MQSHORT
  - MQLONG
  - MQINT8
  - MQUINT8
  - MQINT16
  - MQUINT16
  - MQINT32
  - MQUINT32
  - MQINT64
  - MQUINT64

  Los campos MQCHAR se convierten en página de códigos, pero MQBYTE, MQINT8 y MQUINT8 se dejan intactos. Si la codificación es diferente, MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 y MQBOOL se convierten en consecuencia.
- No utilice los siguientes tipos de datos:
  - double
  - Punteros
  - campos de bits

  Esto se debe a que el programa de utilidad para crear el código de salida de conversión no proporciona el recurso para convertir estos tipos de datos. Para superar esto, puedes escribir tus propias rutinas y llamarlas desde la salida.

Otros puntos a tener en cuenta:

- No utilice números de secuencia en el conjunto de datos de entrada.
- Si hay campos para los que desea proporcionar sus propias rutinas de conversión, declárelos como MQBYTE y, a continuación, sustituya las macros CMQXCFBA generadas por su propio código de conversión.

Ejemplo de C

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR    ID[5];
              MQINT16   VERSION;
              MQBYTE    CODE[4];
              MQLONG    DIMENSIONS[3];
              MQCHAR    NAME[24];
            } ;
```

Esto corresponde a las siguientes declaraciones en otros lenguajes de programación:

COBOL

```
      10 TEST.
       15 SERIAL-NUMBER  PIC S9(9) BINARY.
       15 ID             PIC X(5).
       15 VERSION        PIC S9(4) BINARY.
     * CODE IS NOT TO BE CONVERTED
       15 CODE           PIC X(4).
       15 DIMENSIONS     PIC S9(9) BINARY OCCURS 3 TIMES.
       15 NAME           PIC X(24).
```

System/390

```
TEST           EQU *
SERIAL_NUMBER  DS  F
ID             DS  CL5
VERSION        DS  H
CODE           DS  XL4
DIMENSIONS     DS  3F
NAME           DS  CL24
```

PL/I

Soportado solo en z/OS

```
DCL 1 TEST,
      2 SERIAL_NUMBER  FIXED BIN(31),
      2 ID             CHAR(5),
      2 VERSION        FIXED BIN(15),
      2 CODE           CHAR(4),        /* not to be converted */
      2 DIMENSIONS(3)  FIXED BIN(31),
      2 NAME           CHAR(24);
```

MQ_PUBLISH_EXIT - Salida de publicación

La llamada MQ_PUBLISH_EXIT puede inspeccionar y modificar los mensajes entregados a los suscriptores.

Finalidad

Utilice la salida de publicación para inspeccionar y modificar los mensajes entregados a los suscriptores:

- Examinar el contenido de un mensaje publicado en cada suscriptor.
- Modificar el contenido de un mensaje publicado en cada suscriptor.
- Modificar la cola en la que se coloca el mensaje.
- Detener la entrega de un mensaje a un suscriptor.

Esta salida no está disponible en IBM MQ for z/OS.

Sintaxis

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

Parámetros

***ExitParms* (MQPSXP)- Input/Output**

ExitParms contiene información sobre la invocación de la salida.

***PubContext* (MQPBC) - Input**

PubContext contiene información contextual sobre el editor de la publicación.

***SubContext* (MQSBC) - Input/Output**

SubContext contiene información contextual sobre el suscriptor que recibe la publicación.

MQPSXP-Estructura de datos de salida de publicación

La estructura MQPSXP describe la información que se pasa a y se devuelve de la salida de publicación.

Tabla 818 en la página 1502 resume los campos de la estructura:

Tabla 818. Campos en MQPSXP

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
ExitId	Tipo de salida que se está llamando
ExitReason	Razón para llamar a la salida
ExitResponse	Respuesta de la salida
ExitResponse2	Respuesta secundaria de salida
Feedback	Código de retroalimentación
ExitUserArea	Salir del área de usuario
ExitData	Datos de salida
QMgrName	Nombre del gestor de colas local
Hconn	Descriptor de contexto de conexión
MsgDescPtr	Dirección del descriptor de mensaje (MQMD)
MsgHandle	Descriptor de contexto para propiedades de mensaje (MQHMSG)
MsgInPtr	Dirección del mensaje de entrada
MsgInLength	Longitud del mensaje de entrada
MsgOutPtr	Dirección del mensaje de salida
MsgOutLength	Longitud del mensaje de salida
pEntryPoints	Dirección de la estructura MQIEP

Campos

***StrucID* (MQCHAR4)**

StrucID es el identificador de estructura. El valor es el siguiente:

MQPSXP_STRUCID

MQPSXP_STRUCID es el identificador de la estructura de parámetros de salida de publicación. Para el lenguaje de programación C, la constante MQPSXP_STRUC_ID_ARRAY también está

definida; tiene el mismo valor que MQPSXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

***Version* (MQLONG)**
Version es el número de versión de la estructura. El valor es el siguiente:

MQPSXP_VERSION_1
MQPSXP_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1. La constante MQPSXP_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

***ExitId* (MQLONG)**
ExitId es el tipo de salida que se está llamando. El valor es el siguiente:

MQXT_PUBLISH_EXIT
Salida de publicación.

ExitId es un campo de entrada para la salida.

***ExitReason* (MQLONG)**
ExitReason es la razón por la que se llama a la salida. Los valores posibles son:

MQXR_INIT
Se llama a la salida para esta conexión para la inicialización. La salida puede adquirir e inicializar los recursos que necesita; por ejemplo, el almacenamiento principal.

MQXR_TERM
Se llama a la salida para esta conexión porque la salida está a punto de detenerse. La salida debe liberar los recursos que haya adquirido desde que se inicializó; por ejemplo, el almacenamiento principal.

MQXR_PUBLICATION
El gestor de colas llama a la salida antes de colocar una publicación en una cola de mensajes de un suscriptor. La salida puede cambiar el mensaje, no colocar el mensaje en la cola o detener la publicación.

ExitReason es un campo de entrada para la salida.

***ExitResponse* (MQLONG)**
Establezca *ExitResponse* en la salida para especificar cómo debe continuar el proceso. *ExitResponse* es uno de los valores siguientes:

MQXCC_OK
Establezca MQXCC_OK para continuar procesando normalmente. Establezca MQXCC_OK en respuesta a cualquier valor de ExitReason.

Si ExitReason tiene el valor MQXR_PUBLICATION, los campos *DestinationQName* y *DestinationQMgrName* de la estructura MQSBC identifican el destino al que se envía el mensaje.

MQXCC_FAILED
Establezca MQXCC_FAILED para detener la operación de publicación. El código de terminación MQCC_FAILED y el código de razón 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR se establece en el retorno de la salida.

MQXCC_SUPPRESS_FUNCTION
Establezca MQXCC_SUPPRESS_FUNCTION para detener el proceso normal del mensaje. Solo establezca MQXCC_SUPPRESS_FUNCTION si *ExitReason* tiene el valor MQXR_PUBLICATION.

El gestor de colas sigue procesando el mensaje de acuerdo con la opción MQRO_DISCARD_MSG del campo *Report* del descriptor de mensaje del mensaje.

- Si se especifica la opción MQRO_DISCARD_MSG , el mensaje no se entrega al suscriptor.
- Si no se especifica la opción MQRO_DISCARD_MSG , el mensaje se coloca en la cola de mensajes no entregados. Si no hay ninguna cola de mensajes no entregados o el mensaje no se puede colocar correctamente en la cola de mensajes no entregados, la publicación no se entrega al

suscriptor. La entrega de la publicación a otros suscriptores depende de los valores de los atributos de objeto de tema PMSGDLV y NPMSGDLV . Para obtener una explicación de estos atributos, consulte las descripciones de los parámetros para el mandato DEFINE TOPIC .

ExitResponse es un campo de salida de la salida.

***ExitResponse2* (MQLONG)**
ExitResponse2 está reservado para uso futuro.

***Feedback* (MQLONG)**
Feedback es el código de comentarios que se debe utilizar si la salida devuelve MQXCC_SUPPRESS_FUNCTION en *ExitResponse*.

En la entrada a la salida, *Feedback* siempre tiene el valor MQFB_NONE. Si la salida devuelve MQXCC_SUPPRESS_FUNCTION, establezca *Feedback* en el valor que se utilizará para el mensaje cuando el gestor de colas lo coloque en la cola de mensajes no entregados. Al volver de la salida, si *Feedback* tiene el valor original MQFB_NONE, el gestor de colas establece *Feedback* en MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback es un campo de entrada/salida para la salida.

***ExitUserArea* (MQBYTE16)**
ExitUserArea es un campo que está disponible para que lo utilice la salida. Cada conexión tiene un *ExitUserArea*independiente. La longitud de *ExitUserArea* la proporciona MQ_EXIT_USER_AREA_LENGTH.

El campo *ExitReason* tiene el valor MQXR_INIT en la primera invocación de la salida. *ExitUserArea* se inicializa en MQXUA_NONE en la primera invocación de la salida para una conexión. Los cambios posteriores en *ExitUserArea* se conservan entre invocaciones de la salida.

ExitUserArea es un campo de entrada/salida para la salida.

***ExitData* (MQCHAR32)**
ExitData son datos de salida fijos definidos por el parámetro **PublishExitData** de la stanza en el archivo de inicialización del gestor de colas. Los datos se rellenan con espacios en blanco hasta la longitud completa del campo. Si no hay datos de salida fijos definidos en el archivo de inicialización, *ExitData* está en blanco. La longitud de *ExitData* la proporciona MQ_EXIT_DATA_LENGTH.

ExitData es un campo de entrada para la salida.

***QMgrName* (MQCHAR48)**
QMgrName es el nombre del gestor de colas local. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

QMgrName es un campo de entrada para la salida.

***Hconn* (MQHCONN)**
Hconn es el descriptor de contexto que representa una conexión con el gestor de colas. Utilice sólo *Hconn* como parámetro para las llamadas a la función de propiedad de mensaje MQSETMP, MQINQMMPo MQDLTMP para trabajar con propiedades de mensaje.

Hconn es un campo de entrada para la salida.

***MsgDescPtr* (PMQMD)**
MsgDescPtr es la dirección del descriptor de mensaje (MQMD) del mensaje que se está procesando y es una copia del MQMD devuelto por la llamada MQPUT. La salida puede cambiar el contenido del descriptor de mensaje. Cualquier cambio en el contenido del descriptor de mensaje debe realizarse con cuidado. En concreto, en el caso en el que el campo *SubType* de la estructura MQSBC es de valor MQSUBTYPE_PROXY, el campo *CorrelId* del descriptor de mensaje no debe cambiarse.

No se pasa ningún descriptor de mensaje a la salida si *ExitReason* es MQXR_INIT o MQXR_TERM ; en estos casos, *MsgDescPtr* es el puntero nulo.

MsgDescPtr es un campo de entrada para la salida.

***MsgHandle* (MQHMSG)**
MsgHandle es el descriptor de contexto de las propiedades de mensaje. Utilice sólo *MsgHandle* con las llamadas de función de propiedad de mensaje MQSETMP, MQINQMMPo MQDLTMP para trabajar con propiedades de mensaje.

MsgHandle es un campo de entrada para la salida.

***MsgInPtr* (PMQVOID)**
MsgInPtr es la dirección de los datos del mensaje de entrada. El contenido del almacenamiento intermedio direccionado por *MsgInPtr* puede ser modificado por la salida; consulte *MsgOutPtr* .

MsgInPtr es un campo de entrada para la salida.

***MsgInLength* (MQLONG)**
MsgInLength es la longitud en bytes de los datos de mensaje pasados a la salida. La dirección de los datos la proporciona *MsgInPtr*.

MsgInLength es un campo de entrada para la salida.

***MsgOutPtr* (PMQVOID)**
MsgOutPtr es la dirección de un almacenamiento intermedio que contiene datos de mensaje que se devuelven de la salida. En la entrada a la salida, *MsgOutPtr* es nulo. Al volver de la salida, si el valor sigue siendo nulo, el gestor de colas envía el mensaje especificado por *MsgInPtr*, con la longitud proporcionada por *MsgInLength*.

Si la salida modifica los datos del mensaje, utilice uno de los procedimientos siguientes:

- Si la longitud de los datos no cambia, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr*. En este caso, no cambie *MsgOutPtr* y *MsgOutLength*.
- Si los datos modificados son más cortos que los datos originales, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr*. En este caso, *MsgOutPtr* debe establecerse en la dirección del almacenamiento intermedio de mensajes de entrada y *MsgOutLength* debe establecerse en la nueva longitud de los datos del mensaje.
- Si los datos modificados son, o pueden ser, más largos que los datos originales, la salida debe obtener un nuevo almacenamiento intermedio de mensajes. Copie los datos modificados en él. Establezca *MsgOutPtr* en la dirección del nuevo almacenamiento intermedio y establezca *MsgOutLength* en la longitud de los nuevos datos de mensaje. La salida es responsable de liberar el almacenamiento intermedio direccionado por *MsgOutPtr* cuando se llama a la salida a continuación.

Nota: *MsgOutPtr* es siempre el puntero nulo en la entrada a la salida, y no la dirección de un almacenamiento intermedio de mensajes obtenido anteriormente. Para liberar el almacenamiento intermedio obtenido anteriormente, la salida debe guardar su dirección y longitud. Guarde la información en *ExitUserArea*o en un bloque de control que tenga su dirección guardada en *ExitUserArea*.

MsgOutPtr es un campo de entrada/salida para la salida.

***MsgOutLength* (MQLONG)**
MsgOutLength es la longitud en bytes de los datos de mensaje devueltos por la salida. En la entrada a la salida, este campo es siempre cero. Al volver de la salida, este campo se ignora si *MsgOutPtr* es nulo. Consulte *MsgOutPtr* para obtener información sobre cómo modificar los datos del mensaje.

MsgOutLength es un campo de entrada/salida para la salida.

***pEntryPoints* (PMQIEP)**
pEntryPoints es la dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas MQI y DCI.

Declaración de lenguaje C-MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4    StrucId;             /* Structure identifier */
```

```
    MQLONG     Version;             /* Structure version number */
    MQLONG     ExitId;              /* Type of exit */
    MQLONG     ExitReason;          /* Reason for invoking exit */
    MQLONG     ExitResponse;        /* Response from exit */
    MQLONG     ExitResponse2;       /* Reserved */
    MQLONG     Feedback;            /* Feedback code */
    MQBYTE16   ExitUserArea;        /* Exit user area */
    MQCHAR32   ExitData;            /* Exit data */
    MQCHAR48   QMgrName;            /* Name of local queue manager */
    MQHCONN    Hconn;               /* Connection handle */
    MQHMSG     MsgHandle;           /* Handle to message properties */
    PMQMD      MsgDescPtr;          /* Address of message descriptor */
    PMQVOID    MsgInPtr;            /* Address of input message data */
    MQLONG     MsgInLength;         /* Length of input message data */
    PMQVOID    MsgOutPtr;           /* Address of output message data */
    MQLONG     MsgOutLength;        /* Length of output message data */
    /* Ver:1 */
    PMQIEP     pEntryPoints;        /* Address of the MQIEP structure */
    /* Ver:2 */
}    MQPSXP;
```

MQPBC-Estructura de datos de contexto de publicación

La estructura MQPBC contiene la información contextual, relacionada con el publicador de la publicación, que se pasa a la salida de publicación.

Tabla 819 en la página 1506 resume los campos de la estructura:

Tabla 819. Campos en MQPBC

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
PubTopicString	Serie de tema de publicación
MsgDescPtr	Dirección del descriptor de mensaje (MQMD)

Campos

***StrucID* (MQCHAR4)**

StrucID es el identificador de estructura. El valor es el siguiente:

MQPBC_STRUCID

MQPBC_STRUCID es el identificador de la estructura de contexto de publicación. Para el lenguaje de programación C, la constante MQPBC_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQPBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

***Version* (MQLONG)**

Version es el número de versión de la estructura. El valor es el siguiente:

MQPBC_VERSION_1

MQPBC_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1.

MQPBC_VERSION_2

MQPBC_VERSION_2 es la estructura de parámetros de salida de publicación de la versión 2. La constante MQPBC_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

***PubTopicString* (MQCHARV)**

PubTopicString es la serie de tema en la que se publica.

PubTopicString es un campo de entrada para la salida.

***MsgDescPtr* (PMQMD)**

MsgDescPtr es la dirección de una copia del descriptor de mensaje (MQMD) para el mensaje que se está procesando.

MsgDescPtr es un campo de entrada para la salida.

Declaración de lenguaje C-MQPBC

```
typedef struct tagMQPBC {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQCHARV    PubTopicString;      /* Publish topic string */
  PMQMD      MsgDescPtr;          /* Address of message descriptor */
} MQPBC;
```

MQSBC-Estructura de datos de contexto de suscripción

La estructura MQSBC contiene la información contextual, relacionada con el suscriptor que recibe la publicación, que se pasa a la salida de publicación.

Tabla 820 en la página 1507 resume los campos de la estructura:

Tabla 820. Entrada de campos MQSBC

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
DestinationQMgrName	Nombre del gestor de colas de destino
DestinationQName	Nombre de la cola de destino
SubType	Tipo de suscripción
SubOptions	Opciones de suscripción
ObjectName	Nombre de objeto
ObjectString	Serie de objeto
SubTopicString	Serie de tema de suscripción
SubName	Nombre de suscripción
SubId	Identificador de suscripción
SelectionString	Dirección de la serie de selección
SubLevel	Nivel de suscripción
PSProperties	Propiedades de publicación/suscripción

Campos

***StrucID* (MQCHAR4)**

Identificador de estructura. El valor es el siguiente:

MQSBC_STRUCID

MQSBC_STRUCID es el identificador de la estructura de parámetros de salida de publicación. Para el lenguaje de programación C, la constante MQSBC_STRUC_ID_ARRAY también está definida; MQSBC_STRUC_ID_ARRAY tiene el mismo valor que MQSBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

Version **(MQLONG)**
Número de versión de la estructura. El valor es el siguiente:

MQSBC_VERSION_1
Estructura de parámetros de salida de publicación de la versión 1. La constante MQSBC_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

DestinationQMgrName **(MQCHAR48)**
DestinationQMgrName es el nombre del gestor de colas al que se envía el mensaje. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La salida puede modificar el nombre. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

DestinationQMgrName es un campo de entrada/salida para la salida; consulte la nota.

DestinationQName **(MQCHAR48)**
DestinationQName es el nombre de la cola a la que se envía el mensaje. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La salida puede modificar el nombre. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH.

DestinationQName es un campo de entrada/salida para la salida; consulte la nota.

SubType **(MQLONG)**
SubType indica cómo se ha creado la suscripción. Los valores válidos son MQSUBTYPE_API, MQSUBTYPE_ADMIN y MQSUBTYPE_PROXY ; consulte Consultar estado de suscripción (Respuesta).

SubType es un campo de entrada para la salida.

SubOptions **(MQLONG)**
SubOptions son las opciones de suscripción; consulte "Opciones (MQLONG)" en la página 577 para obtener una descripción de los valores que puede tomar este campo.

SubOptions es un campo de entrada para la salida.

ObjectName **(MQCHAR48)**
ObjectName es el nombre del objeto de tema tal como se define en el gestor de colas local. La longitud de este campo la proporciona MQ_TOPIC_NAME_LENGTH. El nombre de objeto es el nombre del objeto de tema administrativo que el gestor de colas ha asociado con la serie de tema. Incluso si el suscriptor ha proporcionado un objeto de tema como parte de la suscripción, el *ObjectName* puede ser un objeto de tema diferente. La asociación de un objeto de tema con una suscripción depende de la resolución completa de *SubTopicString*.

ObjectName es un campo de entrada para la salida.

ObjectString **(MQCHARV)**
ObjectString es la serie de tema completa de la publicación a la que se ha suscrito. Los comodines de la serie de suscripción original se resuelven. Es diferente al campo MQSD subscription *ObjectString* descrito en "ObjectString (MQCHARV)" en la página 587, que puede contener comodines, y es exclusivo de cualquier nombre de objeto proporcionado por el suscriptor.

ObjectString es un campo de entrada para la salida.

SubTopicString **(MQCHARV)**
SubTopicString es la serie de tema completa tal como la proporciona el suscriptor. *SubTopicString* es la combinación de la serie de tema definida en un objeto de tema y una serie de tema. Un suscriptor debe proporcionar un objeto de tema, una serie de tema o ambos. Si el suscriptor proporciona una serie de tema, puede contener comodines.

SubTopicString es un campo de entrada para la salida.

SubName **(MQCHARV)**
SubName es el nombre de suscripción proporcionado por el suscriptor o es un nombre generado.

SubName es un campo de entrada para la salida.

***SubId* (MQBYTE 24)**
SubId es el identificador de suscripción interno exclusivo.

SubId es un campo de entrada para la salida.

***SelectionString* (MQCHARV)**
SelectionString es el criterio de selección utilizado al suscribirse a mensajes de un tema; consulte Selectores.

SelectionString es un campo de entrada para la salida.

***SubLevel* (MQLONG)**
SubLevel es el nivel de intercepción asociado a la suscripción; consulte "SubLevel (MQLONG)" en la página 591 para obtener más detalles.

SubLevel es un campo de entrada para la salida.

***PSProperties* (MQLONG)**
PSProperties son las propiedades de publicación/suscripción. Especifican cómo se añaden las propiedades de mensaje relacionadas con la publicación/suscripción a los mensajes enviados a esta suscripción. Los valores posibles son MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Consulte Parámetros opcionales (Cambiar, Copiar y Crear suscripción) para obtener una descripción de estos valores.

PSProperties es un campo de entrada para la salida.

Nota: Las comprobaciones de autorización sólo se realizan en los valores originales de *DestinationQMgrName* y *DestinationQName* antes de que se pasen a la salida de publicación. No se realizan nuevas comprobaciones de autorización cuando la salida cambia la cola de destino, ya sea cambiando *DestinationQMgrName* o *DestinationQName*.

Declaración de lenguaje C-MQSBC

```
typedef struct tagMQSBC {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQCHAR48   DestinationQMgrName; /* Destination queue manager */
  MQCHAR48   DestinationQName;    /* Destination queue name */
  MQLONG     SubType;             /* Type of subscription */
  MQLONG     SubOptions;          /* Subscription options */
  MQCHAR48   ObjectName;          /* Object name */
  MQCHARV    ObjectString;        /* Object string */
  MQCHARV    SubTopicString;      /* Subscription topic string */
  MQCHARV    SubName;             /* Subscription name */
  MQBYTE24   SubId;               /* Subscription identifier */
  MQCHARV    SelectionString;     /* Subscription selection string */
  MQLONG     SubLevel;            /* Subscription level */
  MQLONG     PSProperties;        /* Publish/subscribe properties */
} MQSBC;
```

Llamadas de salida de canal y estructuras de datos

Esta colección de temas proporciona información de referencia sobre las llamadas y estructuras de datos especiales de IBM MQ que puede utilizar al escribir programas de salida de canal.

Esta información es información de interfaz de programación sensible al producto. Puede escribir salidas de usuario de IBM MQ en los siguientes lenguajes de programación:

Tabla 821. Salidas de usuario de IBM MQ : plataformas y lenguajes de programación

Plataforma	Lenguajes de programación
IBM MQ for z/OS	Assembler y C (que deben ajustarse al entorno de programación del sistema C para salidas del sistema, descrito en la publicación *z/OS C/C++ Programming Guide*.)

Tabla 821. Salidas de usuario de IBM MQ : plataformas y lenguajes de programación (continuación)	
Plataforma	**Lenguajes de programación**
IBM MQ for IBM i	ILE C, ILE COBOL e ILE RPG
Todas las demás plataformas IBM MQ	C

También puede escribir salidas de usuario en Java para utilizarlas únicamente con aplicaciones Java y JMS . Para obtener más información sobre cómo crear y utilizar salidas de canal con IBM MQ classes for Java, consulte Utilización de salidas de canal en IBM MQ classes for Java y para IBM MQ classes for JMS, consulte Utilización de salidas de canal con IBM MQ classes for JMS.

No puede escribir salidas de usuario de IBM MQ en TAL o Visual Basic. Sin embargo, se proporciona una declaración para la estructura MQCD en Visual Basic para utilizarla en la llamada MQCONNX desde un programa IBM MQ MQI client .

En una serie de casos en las descripciones siguientes, los parámetros son matrices o series de caracteres con un tamaño que no es fijo. Para estos parámetros, se utiliza una "n" en minúsculas para representar una constante numérica. Cuando la declaración para ese parámetro está codificada, "n" debe sustituirse por el valor numérico necesario. Para obtener más información sobre los convenios utilizados en estas descripciones, consulte la publicación “Tipos de datos elementales” en la página 235.

archivos de definición de datos

Los archivos de definición de datos se proporcionan con IBM MQ para cada uno de los lenguajes de programación soportados. Para obtener detalles de estos archivos, consulte Copiar, cabecera, incluir y archivos de módulo.

MQ_CHANNEL_EXIT-Salida de canal

La llamada MQ_CHANNEL_EXIT describe los parámetros que se pasan a cada una de las salidas de canal llamadas por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_EXIT; el nombre MQ_CHANNEL_EXIT no tiene ninguna significación especial puesto que los nombres de las salidas de canal se proporcionan en la definición de canal MQCD.

Hay cinco tipos de salida de canal:

- Salida de seguridad de canal
- Salida de mensajes de canal
- Salida de envío de canal
- Salida de recepción de canal
- Salida de reintento de mensajes de canal

Los parámetros son similares para cada tipo de salida, y la descripción que se proporciona aquí se aplica a todos ellos, excepto cuando se indique específicamente.

Sintaxis

MQ_CHANNEL_EXIT *(**ChannelExitParms, ChannelDefinition, DataLength,***
AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)

Parámetros

La llamada MQ_CHANNEL_EXIT tiene los parámetros siguientes.

Parámetros de ChannelExit(MQCXP)-entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento del canal.

DataLength (MQLONG)-entrada/salida

Longitud de los datos.

Los datos dependen del tipo de salida:

- Para una salida de seguridad de canal, cuando se invoca la salida, este parámetro contiene la longitud de cualquier mensaje de seguridad en el campo *AgentBuffer* , si *ExitReason* es MQXR_SEC_MSG. Es cero si no hay ningún mensaje. La salida debe establecer este campo en la longitud de cualquier mensaje de seguridad que se vaya a enviar a su asociado si establece *ExitResponse* en MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG. Los datos del mensaje se encuentran en *AgentBuffer* o *ExitBufferAddr*.

  El contenido de los mensajes de seguridad es responsabilidad exclusiva de las salidas de seguridad.
- Para una salida de mensajes de canal, cuando se invoca la salida, este parámetro contiene la longitud del mensaje (incluida la cabecera de cola de transmisión). La salida debe establecer este campo en la longitud del mensaje en *AgentBuffer* o *ExitBufferAddr* que va a continuar. Debe ser mayor o igual que la longitud de la cabecera de cola de transmisión (MQXQH).
- Para una salida de emisión o recepción de canal, cuando se invoca la salida, este parámetro contiene la longitud de la transmisión. La salida debe establecer este campo en la longitud de la transmisión en *AgentBuffer* o *ExitBufferAddr* que va a continuar.

Si una salida de seguridad envía un mensaje, y no hay ninguna salida de seguridad en el otro extremo del canal, o el otro extremo establece un *ExitResponse* de MQXCC_OK, la salida de inicio se vuelve a invocar con MQXR_SEC_MSG y una respuesta nula (*DataLength* = 0).

AgentBufferLongitud (MQLONG)-entrada

Longitud del almacenamiento intermedio del agente.

Este parámetro puede ser mayor que *DataLength* en la invocación.

Para las salidas de envío y recepción de mensajes de canal, la salida puede utilizar cualquier espacio no utilizado en la invocación para expandir los datos en su lugar. Si se hace esto, la salida debe establecer correctamente el parámetro **DataLength** .

En el lenguaje de programación C, este parámetro se pasa por dirección.

AgentBuffer (MQBYTE x AgentBufferLength)-entrada/salida

Almacenamiento intermedio de agente.

El contenido de este parámetro depende del tipo de salida:

- Para una salida de seguridad de canal, al invocar la salida contiene un mensaje de seguridad si *ExitReason* es MQXR_SEC_MSG. Para devolver un mensaje de seguridad, la salida puede utilizar este almacenamiento intermedio o su propio almacenamiento intermedio (*ExitBufferAddr*).
- Para una salida de mensajes de canal, al invocar la salida, este parámetro contiene:
  - La cabecera de cola de transmisión (MQXQH), que incluye el descriptor de mensaje (que a su vez contiene la información de contexto del mensaje), seguida inmediatamente de
  - Los datos del mensaje

  Si el mensaje va a continuar, la salida puede realizar una de las acciones siguientes:
  - Deje el contenido del almacenamiento intermedio intacto

- Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength* ; no debe ser mayor que *AgentBufferLength*)
- Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

Los cambios que realiza la salida en la cabecera de cola de transmisión no se comprueban; sin embargo, las modificaciones erróneas pueden significar que el mensaje no se puede colocar en el destino.

- Para una salida de emisión o recepción de canal, al invocar la salida contiene los datos de transmisión. La salida puede realizar una de las acciones siguientes:
  - Deje el contenido del almacenamiento intermedio intacto
  - Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength* ; no debe ser mayor que *AgentBufferLength*)
  - Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

  La salida no debe cambiar los primeros 8 bytes de los datos.

ExitBufferLongitud (MQLONG)-entrada/salida

Longitud del almacenamiento intermedio de salida.

En la primera invocación de la salida, este parámetro se establece en cero. A partir de entonces, cualquier valor que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque. El MCA no utiliza el valor.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no soportan el tipo de datos de puntero.

ExitBufferAddr (MQPTR)-entrada/salida

Dirección del almacenamiento intermedio de salida.

Este parámetro es un puntero a la dirección de un almacenamiento intermedio de almacenamiento gestionado por la salida, donde puede elegir devolver datos de mensaje o transmisión (en función del tipo de salida) al agente si el almacenamiento intermedio del agente es o no es lo suficientemente grande, o si es más conveniente que la salida lo haga.

En la primera invocación de la salida, la dirección pasada a la salida es nula. A partir de entonces, cualquier dirección que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque.

Si ExitBufferAddr es nulo, los datos utilizados se toman del parámetro AgentBuffer .

Si ExitBufferAddr no es nulo, los datos utilizados se toman del almacenamiento intermedio al que apunta el parámetro ExitBufferAddr.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no dan soporte al tipo de datos de puntero.

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition,
&DataLength, &AgentBufferLength, AgentBuffer,
&ExitBufferLength, &ExitBufferAddr);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQCXP   ChannelExitParms;   /* Channel exit parameter block */
MQCD    ChannelDefinition;  /* Channel definition */
MQLONG  DataLength;         /* Length of data */
MQLONG  AgentBufferLength;  /* Length of agent buffer */
MQBYTE  AgentBuffer[n];     /* Agent buffer */
MQLONG  ExitBufferLength;   /* Length of exit buffer */
MQPTR   ExitBufferAddr;     /* Address of exit buffer */
```

Invocación en COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                           DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                           EXITBUFFERLENGTH, EXITBUFFERADDR.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
**   Length of data
 01  DATALENGTH         PIC S9(9) BINARY.
**   Length of agent buffer
 01  AGENTBUFFERLENGTH  PIC S9(9) BINARY.
**   Agent buffer
 01  AGENTBUFFER        PIC X(n).
**   Length of exit buffer
 01  EXITBUFFERLENGTH   PIC S9(9) BINARY.
**   Address of exit buffer
 01  EXITBUFFERADDR     POINTER.
```

Invocación de RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD : DATLEN :
     C                                       ABUFL : ABUF : EBUFL :
     C                                       EBUF)
```

La definición de prototipo para la llamada es:

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     Dexitname         PR                  EXTPROC('exitname')
     D* Channel exit parameter block
     D MQCXP                        160A
     D* Channel definition
     D MQCD                        1328A
     D* Length of data
     D DATLEN                        10I 0
     D* Length of agent buffer
     D ABUFL                         10I 0
     D* Agent buffer
     D ABUF                            *   VALUE
     D* Length of exit buffer
     D EBUFL                         10I 0
     D* Address of exit buffer
     D EBUF                            *
```

Invocación de ensamblador System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X
               AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH,         X
               EXITBUFFERADDR)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
CHANNELEXITPARMS   CMQCXPA  ,      Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,      Channel definition
DATALENGTH         DS       F      Length of data
AGENTBUFFERLENGTH  DS       F      Length of agent buffer
AGENTBUFFER        DS       CL(n)  Agent buffer
EXITBUFFERLENGTH   DS       F      Length of exit buffer
EXITBUFFERADDR     DS       F      Address of exit buffer
```

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCXP.
2. El parámetro **ChannelDefinition** pasado a la salida de canal puede ser una de varias versiones. Consulte el campo *Version* en la estructura MQCD para obtener más información.
3. Si la salida de canal recibe una estructura MQCD con el campo *Version* establecido en un valor mayor que MQCD_VERSION_1, la salida debe utilizar el campo *ConnectionName* en MQCD, en preferencia al campo *ShortConnectionName* .
4. En general, las salidas de canal pueden cambiar la longitud de los datos de mensaje. Esto puede surgir como resultado de que la salida añada datos al mensaje, o elimine datos del mensaje, o comprima o cifre el mensaje. Sin embargo, se aplican restricciones especiales si el mensaje es un segmento que sólo contiene parte de un mensaje lógico. En particular, no debe haber ningún cambio neto en la longitud del mensaje como resultado de las acciones de salidas complementarias de envío y recepción.

   Por ejemplo, se permite que una salida de envío acorte el mensaje comprimiéndolo, pero la salida de recepción complementaria debe restaurar la longitud original del mensaje descomprimiéndolo, de modo que no haya ningún cambio neto en la longitud del mensaje.

   Esta restricción surge porque cambiar la longitud de un segmento haría que los desplazamientos de segmentos posteriores del mensaje fueran incorrectos, y esto inhibiría la capacidad del gestor de colas de reconocer que los segmentos formaban un mensaje lógico completo.

MQ_CHANNEL_AUTO_DEF_EXIT-Salida de definición automática de canal

La llamada MQ_CHANNEL_AUTO_DEF_EXIT describe los parámetros que se pasan a la salida de definición automática de canal llamada por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_AUTO_DEF_EXIT; el nombre MQ_CHANNEL_AUTO_DEF_EXIT no tiene ninguna importancia especial porque los nombres de las salidas de definición automática se proporcionan en el gestor de colas.

Sintaxis

MQ_CHANNEL_AUTO_DEF_EXIT ***(ChannelExitParms, ChannelDefinition)***

Parámetros

La llamada MQ_CHANNEL_AUTO_DEF_EXIT tiene los parámetros siguientes.

Parámetros de ChannelExit(MQCXP)-entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida.La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento de los canales que se crean automáticamente.La salida establece información en esta estructura para modificar el comportamiento predeterminado establecido por el administrador.

La salida no debe modificar los campos MQCD listados:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Si se cambian otros campos, el valor establecido por la salida debe ser válido. Si el valor no es válido, se graba un mensaje de error en el archivo de registro de errores o se visualiza en la consola (según corresponda al entorno).

Atención: Los canales autodefinidos creados mediante una salida de definición automática de canal (CHAD) no pueden establecer la etiqueta de certificado, ya que el reconocimiento TLS se produce en el momento en que se crea el canal. Establecer la etiqueta de certificado en una salida CHAD para los canales de entrada no tiene ningún efecto.

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQCXP  ChannelExitParms;   /* Channel exit parameter block */
MQCD   ChannelDefinition;  /* Channel definition */
```

Invocación en COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
```

Invocación de RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD)
```

La definición de prototipo para la llamada es:

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     Dexitname         PR                EXTPROC('exitname')
     D* Channel exit parameter block
     D MQCXP                      160A
     D* Channel definition
     D MQCD                      1328A
```

Invocación de ensamblador System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
CHANNELEXITPARMS   CMQCXPA  ,  Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,  Channel definition
```

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCXP.
2. El parámetro **ChannelExitParms** pasado a la salida de definición automática de canal es una estructura MQCXP. La versión de MQCXP pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en "MQCXP-Parámetro de salida de canal" en la página 1559 para obtener más detalles.
3. El parámetro **ChannelDefinition** pasado a la salida de definición automática de canal es una estructura MQCD. La versión de MQCD pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en "MQCD-Definición de canal" en la página 1517 para obtener más detalles.

MQXWAIT-Esperar en salida

La llamada MQXWAIT espera a que se produzca un suceso. Sólo se puede utilizar desde una salida de canal en z/OS.

El uso de MQXWAIT ayuda a evitar problemas de rendimiento que de otro modo podrían producirse si una salida de canal hace algo que provoca una espera. El suceso MQXWAIT está en espera es señalado por un MVS ECB (bloque de control de sucesos). El ECB se describe en la descripción del bloque de control MQXWD.

z/OS Para obtener más información sobre el uso de MQXWAIT y la grabación de programas de salida de canal, consulte Escritura de programas de salida de canal en z/OS

Sintaxis

MQXWAIT ***(Hconn, WaitDesc, CompCode, Reason)***

Parámetros

La llamada MQXWAIT tiene los parámetros siguientes.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN anterior emitida en la misma invocación o anterior de la salida.

WaitDesc (MQXWD)-entrada/salida

Descriptor de espera.

Este parámetro describe el suceso a esperar. Consulte "MQXWD-Descriptor de espera de salida" en la página 1574 para obtener detalles de los campos de esta estructura.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los códigos siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_XWAIT_CANCELADO
(2107, X'83B') Se ha cancelado la llamada MQXWAIT.

MQRC_XWAIT_ERROR
(2108, X'83C') Invocación de llamada MQXWAIT no válida.

Invocación en C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQXWD    WaitDesc;  /* Wait descriptor */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación de ensamblador System/390

```
         CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS       F  Connection handle
WAITDESC  CMQXWDA  ,  Wait descriptor
COMPCODE  DS       F  Completion code
REASON    DS       F  Reason code qualifying COMPCODE
```

MQCD-Definición de canal

La estructura MQCD contiene los parámetros que controlan la ejecución de un canal. Se pasa a cada salida de canal que se llama desde un agente de canal de mensajes (MCA).

Si desea más información sobre las salidas de canal, consulte “MQ_CHANNEL_EXIT-Salida de canal” en la página 1510. La descripción de este tema está relacionada con los canales de mensajes y con los canales MQI.

Campos de nombre de salida

Cuando se llama a una salida, el campo relevante de *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit*y *MsgRetryExit* contiene el nombre de la salida que se está invocando actualmente. El significado del nombre en estos campos depende del entorno en el que se ejecuta el MCA. Excepto donde se indique, el nombre se alinea a la izquierda dentro del campo, sin blancos intercalados; el nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

UNIX
El nombre de salida es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[ path ] library ( function )
```

El nombre está limitado a un máximo de 128 caracteres.

z/OS
El nombre de salida es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD. El nombre está limitado a un máximo de ocho caracteres.

Windows
El nombre de salida es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio y una unidad:

```
[d:][ path ] library ( function )
```

El nombre está limitado a un máximo de 128 caracteres.

IBM i
El nombre de salida es un nombre de programa de 10 bytes seguido de un nombre de biblioteca de 10 bytes. Si los nombres tienen menos de 10 bytes de longitud, cada nombre se rellena con espacios en blanco para que tenga 10 bytes. El nombre de biblioteca puede ser `*LIBL` excepto cuando se llama a una salida de definición automática de canal, en cuyo caso se necesita un nombre completo.

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. El valor cambiado permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal. El MQCD modificado también lo utiliza el MCA para su proceso normal durante el tiempo de vida continuo del canal.

La salida no debe modificar los siguientes campos MQCD:

- ChannelName
- ChannelType
- StrucLength
- Versión

Referencia relacionada

“Campos” en la página 1519
Este tema lista todos los campos de la estructura MQCD y describe cada campo.

“Declaración C” en la página 1546
Esta declaración es la declaración C para la estructura MQCD.

“declaración COBOL” en la página 1548
Esta declaración es la declaración COBOL para la estructura MQCD.

“Declaración RPG (ILE)” en la página 1550
Esta declaración es la declaración RPG para la estructura MQCD.

“Declaración de ensamblador System/390” en la página 1553
Esta declaración es la declaración de ensamblador System/390 para la estructura MQCD.

“Declaración de Visual Basic” en la página 1555
Esta declaración es la declaración de Visual Basic de la estructura MQCD.

“Cambio de campos MQCD en una salida de canal” en la página 1556
Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Campos

Este tema lista todos los campos de la estructura MQCD y describe cada campo.

BatchDataLimit (MQLONG)
Este campo especifica el límite, en kilobytes, de la cantidad de datos que se pueden enviar a través de un canal antes de tomar un punto de sincronización.

Un punto de sincronización se alcanza después de que el mensaje que haya provocado que se llegue al límite, haya fluido a través del canal.

El proceso por lotes finaliza cuando se cumple una de las condiciones siguientes:

- Se han enviado **BatchSize** mensajes.
- Se han enviado **BatchDataLimit** bytes.
- La cola de transmisión está vacía y se ha sobrepasado **BatchInterval** .

El valor debe estar en el rango de 0 a 999999. El valor predeterminado es 5000.

Un valor de cero en este atributo significa que no se aplica ningún límite de datos a los lotes a través de este canal.

Este parámetro sólo se aplica a canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSRCVR o MQCHT_CLUSSDR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_11.

BatchHeartbeat (MQLONG)
Este campo especifica el intervalo de tiempo que se utiliza para desencadenar una pulsación por lotes para el canal.

La pulsación por lotes permite a los canales emisores determinar si la instancia de canal remoto sigue activa antes de pasar a ser dudosa. Se produce una pulsación por lotes si un canal emisor no se ha comunicado con la instancia de canal remoto dentro del intervalo de tiempo especificado.

El valor está en el rango de 0 a 999 999; las unidades son milisegundos. Un valor de cero indica que la pulsación por lotes no está habilitada.

Este campo sólo es relevante para los canales que tienen un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

BatchInterval (MQLONG)
Este campo especifica el tiempo aproximado en milisegundos que un canal mantiene abierto un lote, si se han transmitido menos de *BatchSize* mensajes en el lote actual.

Si *BatchInterval* es mayor que cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- Han transcurrido *BatchInterval* milisegundos desde el inicio del lote.

Si *BatchInterval* es cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- la cola de transmisión pasa a estar vacía.

BatchInterval debe estar en el rango de cero a 999 999 999 999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente cuando *Version* es menor que MQCD_VERSION_4.

BatchSize (MQLONG)
Este campo especifica el número máximo de mensajes que se pueden enviar a través de un canal antes de sincronizar el canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

CertificateLabel (MQCHAR64)
Este campo proporciona detalles de la etiqueta de certificado que se está utilizando.

IBM MQ inicializa el valor por omisión para el campo *CertificateLabel* como espacios en blanco.

Esto se interpreta en tiempo de ejecución como el valor predeterminado y es compatible con versiones anteriores.

Por ejemplo, especificar una versión de MQCD menor que 11, o utilizar el valor predeterminado de blancos para el campo *CertificateLabel* , significa que este campo se ignora.

La longitud de este campo la proporciona MQ_CERT_LABEL_LENGTH.

ChannelMonitoring (MQLONG)
Este campo especifica el nivel actual de recopilación de datos de supervisión para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelName (MQCHAR20)
Este campo especifica el nombre de definición de canal.

Debe haber una definición de canal del mismo nombre en la máquina remota para poder comunicarse.

El nombre debe utilizar sólo los caracteres:

- Mayúsculas A-Z
- Minúsculas a-z
- Números 0-9
- Punto (.)
- Barra inclinada (/)
- Subrayado (_)
- Signo de porcentaje (%)

y debe rellenarse a la derecha con espacios en blanco. No se permiten los espacios en blanco iniciales o intercalados.

La longitud de este campo la proporciona MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)
Este campo especifica el nivel actual de recopilación de datos de estadísticas para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelType (MQLONG)
Este campo especifica el tipo de canal.

Es uno de los valores siguientes:

MQCHT_SENDER
Remitente.

MQCHT_SERVER
Servidor.

MQCHT_RECEIVER
Receptor.

MQCHT_REQUESTER
Solicitante.

MQCHT_CLNTCONN
Conexión de cliente.

MQCHT_SVRCONN
Conexión de servidor (para que lo utilicen los clientes).

MQCHT_CLUSSDR
Remitente de clúster.

MQCHT_CLUSRCVR
Receptor de clúster.

Peso de ClientChannel(MQLONG)
Este campo especifica una ponderación para influir en qué definición de canal de conexión de cliente se utiliza.

El atributo de peso ClientChannelse utiliza para que las definiciones de canal de cliente se puedan seleccionar de forma aleatoria en función de su ponderación cuando haya más de una definición adecuada disponible. Cuando un cliente emite una MQCONN solicitando conexión con un grupo de gestores de colas, especificando un nombre de gestor de colas que empieza con un asterisco, y hay más de una definición de canal adecuada disponible en la tabla de definiciones de canal de cliente (CCDT), la definición que se debe utilizar se selecciona aleatoriamente basándose en la ponderación, con cualquier definición de ClientChannelWeight (0) aplicable seleccionada primero en orden alfabético.

Especifique un valor entre 0 y 99. El valor predeterminado es 0.

El valor 0 indica que no se realiza ningún equilibrio de carga y que las definiciones aplicables se seleccionan en orden alfabético. Para habilitar el equilibrio de carga, elija un valor entre 1 y 99, donde 1 es el peso más bajo y 99 el más alto. La distribución de mensajes entre dos o más canales con ponderaciones distintas de cero es proporcional a la proporción de esas ponderaciones. Por ejemplo, tres canales con valores de peso de ClientChannelde 2, 4 y 14 se seleccionan aproximadamente 10%, 20% y 70% del tiempo. Esta distribución no está garantizada.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ClusterPtr (MQPTR)
Este campo especifica la dirección de una lista de nombres de clúster.

Si *ClustersDefined* es mayor que cero, esta dirección es la dirección de una lista de nombres de clúster. El canal pertenece a cada clúster listado.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

ClustersDefined (MQLONG)
Este campo especifica el número de clústeres a los que pertenece el canal.

Este campo es el número de nombres de clúster a los que apunta *ClusterPtr*. Es cero o mayor.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)
Este campo especifica la prioridad del canal de carga de trabajo del clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con la prioridad más alta del conjunto de destinos seleccionados en función del rango. Si hay dos gestores de colas de destino posibles, este atributo se puede utilizar para realizar una migración tras error de un gestor de colas en el otro gestor de colas. Todos los mensajes van al gestor de colas con la prioridad más alta hasta que finaliza, a continuación, los mensajes van al gestor de colas con la siguiente prioridad más alta.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas.

CLWLChannelRank (MQLONG)
Este campo especifica el rango de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con el rango más alto. Cuando el destino final es un gestor de colas en un clúster diferente, puede establecer el rango de gestores de colas de pasarela intermedios (en la intersección de clústeres vecinos) para que el algoritmo de selección elija correctamente un gestor de colas de destino más próximo al destino final.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas.

CLWLChannelWeight (MQLONG)
Este campo especifica el peso del canal de carga de trabajo del clúster.

Peso de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo utiliza el atributo "weight" del canal al desvío de la opción de destino para que se puedan enviar más mensajes a una máquina determinada. Por ejemplo, puede dar a un canal en un servidor UNIX grande un "peso" mayor que otro canal en un PC de escritorio pequeño, y el algoritmo de selección elige el servidor UNIX con más frecuencia que el PC.

El valor está en el rango de 1 a 99. El valor predeterminado es 50.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas.

ConnectionAffinity (MQLONG)
Este campo especifica si las aplicaciones cliente que se conectan varias veces utilizando el mismo nombre de gestor de colas, utilizan el mismo canal de cliente.

Utilice este atributo cuando hay disponibles varias definiciones de canal aplicables.

El valor puede ser uno de los siguientes:

MQCAFTY_PREFERIDO
La primera conexión en un proceso que lee una tabla de definiciones de canal de cliente (CCDT) crea una lista de definiciones aplicables basadas en la ponderación con cualquier definición de CLNTWGHT (0) aplicable primero y en orden alfabético. Cada conexión del proceso intenta conectar utilizando la primera definición de la lista. Si una conexión no es satisfactoria, se utiliza la siguiente definición. Las definiciones no satisfactorias con valores CLNTWGHT distintos de 0 se mueven al final de la lista. Las definiciones CLNTWGHT(0) permanecen en el principio de la lista y se seleccionan en primer lugar para cada conexión.

Cada proceso de cliente con el mismo nombre de host siempre crea la misma lista.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación de .NET (incluida la .NETtotalmente gestionada), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este es el valor predeterminado.

MQCAFTY_NONE
La primera conexión de un proceso que lee una CCDT crea una lista de definiciones aplicables. Todas las conexiones en un proceso seleccionan una definición aplicable según el peso con cualquier definición CLNTWGHT(0) aplicable seleccionada primero en orden alfabético.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación de .NET (incluida la .NETtotalmente gestionada), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ConnectionName (MQCHAR264)
Este campo especifica el nombre de conexión para el canal.

Para los canales de clúster receptor (cuando se especifica), CONNAME está relacionado con el gestor de colas local y para otros canales está relacionado con el gestor de colas de destino. El valor que especifique depende del protocolo de transmisión (*TransportType*) que se va a utilizar:

- Para MQXPT_LU62, es el nombre completo de la unidad lógica asociada.
- Para MQXPT_NETBIOS, es el nombre NetBIOS definido en la máquina remota.
- Para MQXPT_TCP, es el nombre de host, la dirección de red de la máquina remota especificada en formato IPv4 decimal con puntos o IPv6 hexadecimal, o la máquina local para canales de clúster receptor.
- Para MQXPT_SPX, es una dirección de estilo SPX que consta de una dirección de red de 4 bytes, una dirección de nodo de 6 bytes y un número de socket de 2 bytes.

Al definir un canal, este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_RECEIVER. Sin embargo, cuando la definición de canal se pasa a una salida, este campo contiene la dirección del socio, sea cual sea el tipo de canal.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

DataConversion (MQLONG)
Este campo especifica si el agente de canal de mensajes emisor intenta la conversión de los datos de mensaje de aplicación si el agente de canal de mensajes receptor no puede realizar esta conversión.

Este campo sólo se aplica a los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir mensajes que son segmentos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQCDC_SENDER_CONVERSION
Conversión por remitente.

MQCDC_NO_SENDER_CONVERSION
Sin conversión por remitente.

DefReconnect (MQLONG)
El atributo de canal `DefReconnect` establece el valor de atributo de reconexión predeterminado para un canal de conexión de cliente.

La opción de reconexión de cliente automática predeterminada. Puede configurar un IBM MQ MQI client para que reconecte automáticamente una aplicación cliente. El IBM MQ MQI client intenta reconectarse a un gestor de colas después de la anomalía de una anomalía de la conexión. Intenta reconectarse sin que el cliente de la aplicación emita una llamada `MQCONN` o `MQCONNX` MQI.

La reconexión es una opción de `MQCONNX` . Utilizando el atributo de canal `DefReconnect` puede añadir un comportamiento de reconexión a las aplicaciones existentes que utilizan `MQCONN`. También puede cambiar el comportamiento de reconexión de las aplicaciones que utilizan `MQCONNX`.

También puede establecer el valor `DefRecon` desde el archivo `mqclient.ini` para establecer o modificar el comportamiento de reconexión. El valor `DefRecon` del archivo `mqclient.ini` tiene prioridad sobre el atributo de canal `DefReconnect` .

Syntax

`DefReconnect` (`MQRCN_NO` (default) |`MQRCN_YES`|`MQRCN_Q_MGR`|`MQRCN_DISABLED`)

Parámetros

`MQRCN_NO`
`MQRCN_NO` es el valor predeterminado.

A menos que **`MQCONNX`**lo altere temporalmente, el cliente no se vuelve a conectar automáticamente.

`MQRCN_YES`
A menos que **`MQCONNX`**lo altere temporalmente, el cliente se vuelve a conectar automáticamente.

`MQRCN_Q_MGR`
A menos que lo altere temporalmente **`MQCONNX`**, el cliente se vuelve a conectar automáticamente, pero sólo al mismo gestor de colas. La opción `QMGR` tiene el mismo efecto que `MQCNO_RECONNECT_Q_MGR`.

`MQRCN_DISABLED`
La reconexión está inhabilitada, aunque lo solicite el programa cliente utilizando la llamada MQI de **`MQCONNX`** .

IBM MQ classes for Java no da soporte a la reconexión automática del cliente.

Tabla 822. La reconexión automática depende de los valores establecidos en la aplicación y en la definición de canal

DefReconnect	**Opciones de reconexión establecidas en la aplicación**			
	`MQCNO_RECONNECT`	`MQCNO_RECONNECT_Q_MGR`	`MQCNO_RECONNECT_AS_DEF`	`MQCNO_RECONNECT_DISABLED`
`MQRCN_NO`	SÍ	QMGR	NO	NO
`MQRCN_YES`	SÍ	QMGR	SÍ	NO

Tabla 822. La reconexión automática depende de los valores establecidos en la aplicación y en la definición de canal (continuación)				
DefReconnect	**Opciones de reconexión establecidas en la aplicación**			
MQRCN_Q_MGR	SÍ	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

Conceptos relacionados

Reconexión de cliente automática

Reconexión de canal y cliente

Stanza CHANNELS del archivo de configuración de cliente

Referencia relacionada

"Opciones (MQLONG)" en la página 323
Opciones que controlan la acción de MQCONNX.

Descripción (MQCHAR64)
Este campo se puede utilizar para comentarios descriptivos.

El contenido del campo no es significativo para los agentes de canal de mensajes. Sin embargo, sólo debe contener caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como se define en el atributo de gestor de colas **CodedCharSetId**), es posible que estos caracteres se conviertan incorrectamente si este campo se envía a otro gestor de colas.

La longitud de este campo la proporciona MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)
Este campo especifica el tiempo máximo en segundos durante el cual el canal espera a que llegue un mensaje a la cola de transmisión, antes de terminar el canal.

En otras palabras, especifica el intervalo de desconexión.

El valor A de cero hace que el MCA espere indefinidamente.

Para los canales de conexión de servidor que utilizan el protocolo TCP, el intervalo representa el valor de desconexión de inactividad del cliente, especificado en segundos. Si una conexión de servidor no ha recibido ninguna comunicación de su cliente asociado durante este tiempo, termina la conexión. El intervalo de inactividad de conexión de servidor sólo se aplica entre llamadas de API de IBM MQ desde un cliente, por lo que ningún cliente se desconecta durante una MQGET de larga ejecución con llamada de espera.

Este atributo no es aplicable para los canales de conexión de servidor que utilizan protocolos distintos de TCP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR o MQCHT_SVRCONN.

ExitDataLongitud (MQLONG)
Este campo especifica la longitud en bytes de cada uno de los elementos de datos de usuario en las listas de elementos de datos de usuario de salida a los que se dirigen los campos *MsgUserDataPtr*, *SendUserDataPtr*y *ReceiveUserDataPtr* .

Esta longitud no es necesariamente la misma que MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ExitNameLongitud (MQLONG)
Este campo especifica la longitud en bytes de cada uno de los nombres de las listas de nombres de salida a los que se dirigen los campos *MsgExitPtr*, *SendExitPtr*y *ReceiveExitPtr* .

Esta longitud no es necesariamente la misma que MQ_EXIT_NAME_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Lista de HdrComp[2] (MQLONG)
Este campo especifica la lista de técnicas de compresión de datos de cabecera soportadas por el canal.

La lista contiene uno o varios de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA
Se lleva a cabo la compresión de datos de cabecera.

Los valores no utilizados en la matriz se establecen en MQCOMPRESS_NOT_AVAILABLE.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

HeartbeatInterval (MQLONG)
Este campo especifica el tiempo en segundos entre los flujos de pulsaciones.

La interpretación de este campo depende del tipo de canal, tal como se indica a continuación:

- Para un tipo de canal de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR, este campo es el tiempo en segundos entre los flujos de pulsaciones pasados desde el MCA emisor cuando no hay mensajes en la cola de transmisión. Esto da al MCA receptor la oportunidad de desactivar temporalmente el canal. Para ser útil, *HeartbeatInterval* debe ser menor que *DiscInterval*.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en cero, este campo es el tiempo en segundos entre los flujos de latido pasados desde el MCA del servidor cuando dicho MCA ha emitido una llamada MQGET con la opción MQGMO_WAIT en nombre de una aplicación cliente. Esto permite al MCA del servidor manejar situaciones en las que la conexión de cliente falla durante un MQGET con MQGMO_WAIT.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en un valor distinto de cero, este campo es el tiempo en segundos entre el flujo de pulsaciones cuando no hay flujos de datos enviados o recibidos. Esto permite que el canal se desactive temporalmente de forma eficiente.

El valor está en el rango de 0 a 999 999. El valor que se utiliza es el mayor de los valores especificados en el lado de envío y el lado de recepción a menos que se especifique un valor de 0 en cada lado, en cuyo caso no se produce ningún intercambio de pulsaciones.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Intervalo de KeepAlive(MQLONG)
Este campo especifica el valor pasado a la pila de comunicaciones para la temporización de estado activo para el canal.

El valor es aplicable para los protocolos de comunicaciones TCP/IP y SPX, aunque no todas las implementaciones soportan este parámetro.

El valor está en el rango de 0 a 99 999; las unidades son segundos. Un valor de cero indica que el estado activo del canal no está habilitado, aunque el estado activo puede seguir produciéndose si el estado activo TCP/IP (en lugar del estado activo del canal) está habilitado. También es válido el siguiente valor especial:

MQKAI_AUTO
Automático.

Este valor indica que el intervalo de estado activo se calcula a partir del intervalo de latido negociado, como se indica a continuación:

- Si el intervalo de pulsaciones negociado es mayor que cero, el intervalo de estado activo que se utiliza es el intervalo de pulsaciones más 60 segundos.
- Si el intervalo de pulsaciones negociado es cero, el intervalo de estado activo que se utiliza es cero.

- En z/OS, el estado activo de TCP/IP se produce cuando se especifica TCPKEEP (YES) en el objeto del gestor de colas.
- En otros entornos, el estado activo de TCP/IP se produce cuando se especifica el parámetro **KEEPALIVE=YES** en la stanza TCP del archivo de configuración de gestión de colas distribuidas.

Este campo sólo es relevante para los canales que tienen un *TransportType* de MQXPT_TCP o MQXPT_SPX.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

LocalAddress (MQCHAR48)
Este campo especifica la dirección TCP/IP local definida para el canal para las comunicaciones de salida.

Este campo está en blanco si no se ha definido ninguna dirección específica para las comunicaciones de salida. La dirección puede incluir opcionalmente un número de puerto o un rango de números de puerto. El formato de esta dirección es:

```
[ip-addr][(low-port[,high-port])]
```

donde los corchetes ([]) indican información opcional, `ip-addr` se especifica en formato IPv4 decimal con puntos, IPv6 hexadecimal o alfanumérico, y `low-port` y `high-port` son números de puerto entre paréntesis. Todos son opcionales.

Una dirección IP, puerto o rango de puertos específicos para las comunicaciones de salida es útil en escenarios de recuperación en los que un canal se reinicia en una pila TCP/IP diferente.

LocalAddress es similar en forma a *ConnectionName*, pero no debe confundirse con él. *LocalAddress* especifica las características de las comunicaciones locales, mientras que *ConnectionName* especifica cómo llegar a un gestor de colas remoto.

V 9.1.0.8 A partir de IBM MQ 9.1.0 Fix Pack 8, la JMQI (Java Message Queueing Interface) se ha actualizado para asegurarse de que el campo de dirección local se ha establecido en un objeto MQCD después de que se haya creado una instancia de canal y se haya conectado a un gestor de colas. Esto significa que cuando una salida de canal escrita en Java llama al método `MQCD.getLocalAddress()`, el método devuelve la dirección local que la instancia de canal está utilizando. Antes de IBM MQ 9.1.0 Fix Pack 8, la salida de seguridad de canal no podía acceder a la dirección local utilizada por la instancia de canal y el método `MQCD.getLocalAddress()` devolvía un valor nulo.

Este campo sólo es relevante para canales con un *TransportType* de MQXPT_TCP y un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_LOCAL_ADDRESS_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)
Este campo especifica la longitud en bytes del identificador de usuario MCA completo al que apunta *LongMCAUserIdPtr*.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)
Este campo especifica la dirección del identificador de usuario de MCA largo.

Si *LongMCAUserIdLength* es mayor que cero, este campo es la dirección del identificador de usuario MCA completo. La longitud del identificador completo la proporciona *LongMCAUserIdLength*. Los primeros 12 bytes del identificador de usuario de MCA también están contenidos en el campo *MCAUserIdentifier*.

Consulte la descripción del campo *MCAUserIdentifier* para obtener detalles del identificador de usuario de MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN o MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdLongitud (MQLONG)
Este campo especifica la longitud en bytes del identificador de usuario remoto completo al que apunta *LongRemoteUserIdPtr*.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)
Este campo especifica la dirección del identificador de usuario remoto largo.

Si *LongRemoteUserIdLength* es mayor que cero, este distintivo es la dirección del identificador de usuario remoto completo. La longitud del identificador completo la proporciona *LongRemoteUserIdLength*. Los primeros 12 bytes del identificador de usuario remoto también están contenidos en el campo *RemoteUserIdentifier*.

Consulte la descripción del campo *RemoteUserIdentifier* para obtener detalles del identificador de usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

Recuento de LongRetry(MQLONG)
Este campo especifica el recuento utilizado después de que se haya agotado el recuento especificado por *ShortRetryCount* .

Especifica el número máximo de intentos adicionales que se realizan para conectarse a la máquina remota, a intervalos especificados por *LongRetryInterval*, antes de registrar un error en el operador.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Intervalo LongRetry(MQLONG)
Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

MaxInstances (MQLONG)
Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente, estas instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes, de modo que el número de instancias actualmente en ejecución sea menor que el valor del campo.

MaxInstancesPerClient (MQLONG)
Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar desde un único cliente.

En este contexto, las conexiones que se originan desde la misma dirección de red remota se consideran procedentes del mismo cliente.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente desde clientes individuales, las instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias de cualquiera de estos clientes no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes de modo que el número de instancias actualmente en ejecución, que se originan en el cliente que intenta iniciar una nueva, sea menor que el valor del campo.

MaxMsgLongitud (MQLONG)
Este campo especifica la longitud máxima de mensaje que puede transmitirse en el canal.

Este valor se compara con el del canal remoto y el máximo real es el menor de los dos valores.

MCAName (MQCHAR20)
Este campo es un campo reservado.

El valor de este campo está en blanco.

La longitud de este campo la proporciona MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)
Este campo especifica el identificador de seguridad para el MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

MCAType (MQLONG)
Este campo especifica el tipo de programa de agente de canal de mensajes.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQMCAT_PROCESO
proceso.

El agente de canal de mensajes se ejecuta como un proceso independiente.

HEBRA MQMCAT_THREAD
Hebra (IBM i, UNIXy Windows).

El agente de canal de mensajes se ejecuta como una hebra separada.

Este campo no está presente cuando *Versión* es menor que MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)
Este campo especifica el identificador de usuario para el agente de canal de mensajes (MCA).

Este campo utiliza los primeros 12 bytes del identificador de usuario de MCA y puede ser establecido por un agente de seguridad.

Hay dos campos que contienen el identificador de usuario de MCA:

- *MCAUserIdentifier* contiene los primeros 12 bytes del identificador de usuario de MCA y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *MCAUserIdentifier* puede estar en blanco.
- *LongMCAUserIdPtr* apunta al identificador de usuario de MCA completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongMCAUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongMCAUserIdLength* es cero y el valor de *LongMCAUserIdPtr* no está definido.

  Nota: *LongMCAUserIdPtr* no está presente si *Version* es menor que MQCD_VERSION_6.

Si el identificador de usuario de MCA no está en blanco, especifica el identificador de usuario que utilizará el agente de canal de mensajes para la autorización para acceder a los recursos de IBM MQ . Para los tipos de canal MQCHT_REQUESTER, MQCHT_RECEIVER y MQCHT_CLUSRCVR, si PutAuthority es MQPA_DEFAULT, este es el identificador de usuario utilizado para las comprobaciones de autorización para la operación de colocación en colas de destino.

Si el identificador de usuario de MCA está en blanco, el agente de canal de mensajes utiliza su identificador de usuario predeterminado.

El identificador de usuario de MCA se puede establecer mediante una salida de seguridad para indicar el identificador de usuario que debe utilizar el agente de canal de mensajes. La salida puede cambiar *MCAUserIdentifier*o la serie a la que apunta *LongMCAUserIdPtr*. Si ambos se cambian pero difieren entre sí, el MCA utiliza *LongMCAUserIdPtr* en preferencia a *MCAUserIdentifier*. Si la salida cambia la longitud de la serie a la que se refiere *LongMCAUserIdPtr*, *LongMCAUserIdLength* debe establecerse de forma correspondiente. Si la salida aumenta la longitud del identificador, la salida debe asignar el almacenamiento de la longitud necesaria, establecer dicho almacenamiento en el identificador necesario y colocar la dirección de dicho almacenamiento en *LongMCAUserIdPtr*. La salida es responsable de liberar ese almacenamiento cuando la salida se invoca más tarde con la razón MQXR_TERM.

Para los canales con un *ChannelType* de MQCHT_SVRCONN, si *MCAUserIdentifier* en la definición de canal está en blanco, cualquier identificador de usuario transferido desde el cliente se copia en él. Este identificador de usuario (después de cualquier modificación por parte de la salida de seguridad en el servidor) es el que se supone que ejecuta la aplicación cliente.

El identificador de usuario de MCA no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente cuando *Version* es menor que MQCD_VERSION_2.

ModeName (MQCHAR8)
Este campo especifica el nombre de modalidad de LU 6.2 .

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco. En su lugar, la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_MODE_NAME_LENGTH.

MsgCompLista [16] (MQLONG)
Este campo especifica la lista de técnicas de compresión de datos de mensaje soportadas por el canal.

La lista contiene uno o varios de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de mensaje.

MQCOMPRESS_RLE
Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

Los valores no utilizados en la matriz se establecen en MQCOMPRESS_NOT_AVAILABLE.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

MsgExit (MQCHARn)
Este campo especifica el nombre de salida de mensaje de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de que se haya recuperado un mensaje de la cola de transmisión (emisor o servidor), o inmediatamente antes de que se coloque un mensaje en una cola de destino (receptor o solicitante).

  A la salida se le proporciona toda la cabecera de cola de mensajes y transmisión de la aplicación para su modificación.
- Durante la inicialización y terminación del canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN; nunca se invoca una salida de mensaje para dichos canales.

Consulte "MQCD-Definición de canal" en la página 1517 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

MsgExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *MsgExit* .

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de mensaje de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

MsgExitsdefinido (MQLONG)
Este campo especifica el número de salidas de mensajes de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Recuento de MsgRetry(MQLONG)
Este campo especifica el número de veces que el MCA intenta colocar el mensaje, después de que el primer intento haya fallado.

Este campo indica el número de veces que el MCA intenta la operación de apertura o colocación, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED. El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo **MsgRetryCount** controla si el MCA intenta reintentos. Si el valor del atributo es cero, no se intenta ningún reintento. Si el valor del atributo es mayor que cero, los reintentos se intentan a intervalos proporcionados por el atributo **MsgRetryInterval** .

  Los reintentos sólo se intentan para los siguientes códigos de razón:

  – MQRC_PAGESET_FULL
  – MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
  – MQRC_Q_FULL

  Para otros códigos de razón, el MCA continúa inmediatamente con su proceso de anomalía normal, sin reintentar el mensaje anómalo.
- Si *MsgRetryExit* no está en blanco, el atributo **MsgRetryCount** no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuántas veces se intenta el reintento y a qué intervalos; la salida se invoca aunque el atributo **MsgRetryCount** sea cero.

  El atributo **MsgRetryCount** está disponible para la salida en la estructura MQCD, pero la salida que no es necesaria para respetarlo-los reintentos continúan indefinidamente hasta que la salida devuelve MQXCC_SUPPRESS_FUNCTION en el campo *ExitResponse* de MQCXP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Salida MsgRetry(MQCHARn)
Este campo especifica el nombre de salida de reintento de mensaje de canal.

La salida de reintento de mensaje es una salida que invoca el MCA cuando el MCA recibe un código de terminación de MQCC_FAILED de una llamada MQOPEN o MQPUT. La finalidad de la salida es especificar

un intervalo de tiempo durante el cual el MCA espera antes de volver a intentar la operación MQOPEN o MQPUT. De forma alternativa, la salida se puede establecer para no volver a intentar la operación.

La salida se invoca para todos los códigos de razón que tienen un código de terminación de MQCC_FAILED-los valores de la salida determinan qué códigos de razón desea que el MCA vuelva a intentarlo, para cuántos intentos y en qué intervalos de tiempo.

Cuando la operación ya no se intenta, el MCA realiza su proceso de anomalía normal; este proceso incluye la generación de un mensaje de informe de excepción (si lo especifica el remitente) y la colocación del mensaje original en la cola de mensajes no entregados o el descarte del mensaje (según si el remitente ha especificado MQRO_DEAD_LETTER_Q o MQRO_DISCARD_MSG). Las anomalías relacionadas con la cola de mensajes no entregados (por ejemplo, la cola de mensajes no entregados llena) no hacen que se invoque la salida de reintento de mensaje.

Si el nombre de salida no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de realizar la espera antes de volver a intentar entregar un mensaje
- Durante la inicialización y terminación del canal

Consulte "MQCD-Definición de canal" en la página 1517 para obtener una descripción del contenido de este campo en varios entornos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Intervalo MsgRetry(MQLONG)
Este campo especifica el intervalo mínimo en milisegundos después del cual se reintenta la operación de apertura o colocación.

El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo **MsgRetryInterval** especifica el periodo mínimo que el MCA espera antes de reintentar un mensaje, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED. Un valor de cero significa que el reintento se realizará lo antes posible después del intento anterior. Los reintentos sólo se realizan si *MsgRetryCount* es mayor que cero.

  Este atributo también se utiliza como tiempo de espera si la salida de reintento de mensaje devuelve un valor no válido en el campo *MsgRetryInterval* de MQCXP.
- Si *MsgRetryExit* no está en blanco, el atributo **MsgRetryInterval** no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuánto tiempo espera el MCA. El atributo **MsgRetryInterval** se pone a disposición de la salida en la estructura MQCD, pero la salida no es necesaria para respetarlo.

El valor está en el rango de 0 a 999.999.999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)
Este campo especifica los datos de usuario de salida de reintento de mensaje de canal.

Estos datos se pasan a la salida de reintento de mensaje de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH. Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Este campo no es relevante en IBM MQ for IBM i.

Datos de MsgUser(MQCHAR32)
Este campo especifica los datos de usuario de salida de mensajes de canal.

Estos datos se pasan a la salida de mensajes de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

MsgUserDataPtr (MQPTR)
Este campo especifica la dirección del primer campo *MsgUserData* .

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

NetworkPriority (MQLONG)
Este campo especifica la prioridad de la conexión de red para el canal.

Cuando hay disponibles varias vías de acceso a un destino determinado, se elige la vía de acceso con la prioridad más alta. El valor está en el rango de 0 a 9; 0 es la prioridad más baja.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)
Este campo especifica la velocidad a la que viajan los mensajes no persistentes a través del canal.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQNPMS_NORMAL
Velocidad normal.

Si un canal está definido como MQNPMS_NORMAL, los mensajes no persistentes viajan a través del canal a una velocidad normal. Esto tiene la ventaja de que estos mensajes no se pierden si hay una anomalía de canal. Además, los mensajes persistentes y no persistentes en la misma cola de transmisión mantienen su orden relativo entre sí.

MQNPMS_FAST
Velocidad rápida.

Si un canal está definido como MQNPMS_FAST, los mensajes no persistentes viajan a través del canal a una velocidad rápida. Esto mejora el rendimiento del canal, pero significa que los mensajes no persistentes se pierden si se produce una anomalía de canal. Además, es posible que los mensajes no persistentes salten por delante de los mensajes persistentes que esperan en la misma cola de transmisión, es decir, el orden de los mensajes no persistentes no se mantiene en relación con los mensajes persistentes. Sin embargo, se mantiene el orden de los mensajes no persistentes relativos entre sí. De forma similar, se mantiene el orden de los mensajes persistentes relativos entre sí.

Contraseña (MQCHAR12)
Este campo especifica la contraseña utilizada por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en UNIXy Windows, y sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

PropertyControl (MQLONG)
Este campo especifica qué sucede con las propiedades de los mensajes cuando el mensaje está a punto de enviarse a un gestor de colas V6 o anterior (un gestor de colas que no entiende el concepto de un descriptor de propiedad).

El valor puede ser cualquiera de los valores siguientes:

COMPATIBILIDAD de MQPROP_COMPATIBILITY
Si el mensaje contiene una propiedad con el prefijo **mcd.**, **jms.**, **usr.** o **mqext.**, todas las propiedades del mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión), se descartan y ya no son accesibles para la aplicación.

Este valor es el valor predeterminado; permite que las aplicaciones, que esperan que las propiedades relacionadas con JMSestén en una cabecera MQRFH2 en los datos del mensaje, continúen trabajando sin modificar.

MQPROP_NONE
Todas las propiedades del mensaje, excepto las propiedades del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se envíe al gestor de colas remoto.

MQPROP_ALL
Todas las propiedades del mensaje se incluyen con el mensaje cuando se envía al gestor de colas remoto. Las propiedades, excepto aquellas que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje.

Este atributo es aplicable a los canales Remitente, Servidor, Remitente de clúster y Receptor de clúster.

"MQIA_ * (Selectores de atributos enteros)" en la página 129
"MQPROP_ * (Valores de control de propiedades de cola y canal y longitud máxima de propiedades)" en la página 169

PutAuthority (MQLONG)
Este campo especifica si el identificador de usuario en la información de contexto asociada con un mensaje se utiliza para establecer la autorización para transferir el mensaje a la cola de destino.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQPA_PREDETERMINADO
Se utiliza el identificador de usuario predeterminado.

CONTEXTO_MQPA
Se utiliza el identificador de usuario de contexto.

MQPA_ALTERNATE_OR_MCA
Se utiliza el ID de usuario del campo UserIdentifier del descriptor de mensaje. No se utiliza ningún ID de usuario recibido de la red. Este valor sólo está soportado en z/OS.

MQPA_ONLY_MCA
Se utiliza el ID de usuario predeterminado. No se utiliza ningún ID de usuario recibido de la red. Este valor sólo está soportado en z/OS.

QMgrName (MQCHAR48)
Este campo especifica el nombre del gestor de colas al que se puede conectar una salida.

Para canales con un *ChannelType* distinto de MQCHT_CLNTCONN, este campo es el nombre del gestor de colas al que se puede conectar una salida, que en UNIX, Linux, and Windows, siempre no está en blanco.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)
Este campo especifica el nombre de salida de recepción de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se procesen los datos de red recibidos.

  A la salida se le proporciona el almacenamiento intermedio de transmisión completo tal como se ha recibido. El contenido del almacenamiento intermedio se puede modificar según sea necesario.
- Durante la inicialización y terminación del canal.

Consulte "MQCD-Definición de canal" en la página 1517 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

ReceiveExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *ReceiveExit* .

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de recepción de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ReceiveExitsdefinido (MQLONG)
Este campo especifica el número de salidas de recepción de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Datos de ReceiveUser(MQCHAR32)
Este canal especifica los datos de usuario de salida de recepción de canal.

Estos datos se pasan a la salida de recepción de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)
Este campo especifica la dirección del primer campo *ReceiveUserData* .

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de recepción de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_5.

RemotePassword (MQCHAR12)
Este campo especifica la contraseña de un socio.

Este campo sólo contiene información válida si *ChannelType* es MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, esta contraseña es una contraseña que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener una contraseña que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede utilizar esta contraseña para validar el identificador de usuario en *RemoteUserIdentifier*.

  Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

RemoteSecurityId (MQBYTE40)
Este campo especifica el identificador de seguridad para el usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_7.

Identificador RemoteUser(MQCHAR12)
Este campo especifica los primeros 12 bytes de un identificador de usuario de un socio.

Hay dos campos que contienen el identificador de usuario remoto:

- *RemoteUserIdentifier* contiene los primeros 12 bytes del identificador de usuario remoto y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *RemoteUserIdentifier* puede estar en blanco.
- *LongRemoteUserIdPtr* apunta al identificador de usuario remoto completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongRemoteUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongRemoteUserIdLength* es cero y el valor de *LongRemoteUserIdPtr* no está definido.

  LongRemoteUserIdPtr no está presente si *Version* es menor que MQCD_VERSION_6.

El identificador de usuario remoto sólo es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, este valor es un identificador de usuario que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener un identificador de usuario que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede validar este ID de usuario (posiblemente con la contraseña en *RemotePassword*) y actualizar el valor en *MCAUserIdentifier*.

  Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

SecurityExit (MQCHARn)
Este campo especifica el nombre de salida de seguridad de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de establecer un canal.

  Antes de transferir ningún mensaje, se da la oportunidad a la salida de causar flujos de mensajes de seguridad para validar la autorización de conexión.
- Al recibir una respuesta a un flujo de mensajes de seguridad.

  Los flujos de mensajes de seguridad recibidos del procesador remoto en la máquina remota se asignan a la salida.
- Durante la inicialización y terminación del canal.

Consulte "MQCD-Definición de canal" en la página 1517 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

Datos de SecurityUser(MQCHAR32)
Este canal especifica los datos de usuario de salida de seguridad de canal.

Estos datos se pasan a la salida de seguridad de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

SendExit (MQCHARn)
Este campo especifica el nombre de salida de emisión de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se envíen los datos en la red.

  A la salida se le proporciona el almacenamiento intermedio de transmisión completo antes de que se transmita. El contenido del almacenamiento intermedio se puede modificar según sea necesario.
- Durante la inicialización y terminación del canal.

Consulte "MQCD-Definición de canal" en la página 1517 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

SendExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *SendExit* .

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de envío de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *SendExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de envío de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SendExitsdefinidos (MQLONG)
Este campo especifica el número de salidas de envío de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Datos de SendUser(MQCHAR32)
Este campo especifica los datos de usuario de salida de emisión de canal.

Estos datos se pasan a la salida de emisión de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

SendUserDataPtr (MQPTR)
Este campo especifica la dirección del campo *SendUserData* .

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SeqNumberEnvolver (MQLONG)
Este campo especifica el número de secuencia de mensaje más alto permitido.

Cuando se alcanza este valor, los números de secuencia se reinician para empezar de nuevo en 1.

Este valor no es negociable y debe coincidir en las definiciones de canal local y remoto.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

SharingConversations (MQLONG)
Este campo especifica el número máximo de conversaciones que pueden compartir una instancia de canal asociada con este canal.

Este campo se utiliza en la conexión de cliente y en los canales de conexión de servidor.

Un valor de 0 significa que el canal funciona como lo hacía en versiones anteriores a IBM WebSphere MQ 7.0 con respecto a los atributos siguientes:

- Compartimiento de conversaciones
- Lectura hacia adelante
- STOP CHANNEL(*channelname*) MODE(QUIESCE)
- Pulsaciones
- Consumo asíncrono de cliente

Un valor de 1 es el valor mínimo para el comportamiento de IBM WebSphere MQ 7.0 . Aunque sólo se permite una conversación en la instancia de canal, están disponibles la lectura anticipada, el consumo asíncrono y el comportamiento IBM WebSphere MQ 7.0 de CLNTCONN-SVRCONN la pulsación y la detención de canal inactiva.

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_9.

El valor predeterminado de este campo es 10.

Nota: Los límites *MaxInstances* y *MaxInstancesPerClient* aplicados a un canal restringen el número de instancias de canal, no el número de conversaciones que pueden estar compartiendo dichas instancias.

Nombre de ShortConnection(MQCHAR20)
Este campo especifica los primeros 20 bytes de un nombre de conexión.

Si el campo *Version* es MQCD_VERSION_1, *ShortConnectionName* contiene el nombre de conexión completo.

Si el campo *Version* es MQCD_VERSION_2 o superior, *ShortConnectionName* contiene los primeros 20 caracteres del nombre de conexión. El nombre completo de la conexión se proporciona mediante el campo *ConnectionName* ; *ShortConnectionName* y los primeros 20 caracteres de *ConnectionName* son idénticos.

Consulte *ConnectionName* para obtener detalles del contenido de este campo.

Nota: El nombre de este campo se ha cambiado para MQCD_VERSION_2 y las versiones posteriores de MQCD; el campo se denominaba anteriormente *ConnectionName*.

La longitud de este campo la proporciona MQ_SHORT_CONN_NAME_LENGTH.

Recuento de ShortRetry(MQLONG)
Este campo especifica el número máximo de intentos que se realizan para conectarse a una máquina remota.

Este campo es el número máximo de intentos que se realizan para conectarse a la máquina remota, a intervalos especificados por *ShortRetryInterval*, antes de que se utilicen *LongRetryCount* y *LongRetryInterval* (normalmente más largos).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

ShortRetryIntervalo (MQLONG)
Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

z/OS V 9.1.3 *SPLProtection (MQLONG)*
Este campo especifica el valor de la protección de política de seguridad de AMS .

El valor puede ser uno de los siguientes:

MQSPL_PASSTHRU
Pase a través, sin cambios, los mensajes enviados o recibidos por el MCA para este canal.

Este valor sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER o MQCHT_REQUESTER, y es el valor predeterminado.

MQSPL_ELIMINAR
Elimine cualquier protección AMS de los mensajes recuperados de la cola de transmisión por el MCA y envíe los mensajes al asociado.

Este valor sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER o MQCHT_SERVER.

MQSPL_ASPOLICY
Basándose en la política definida para la cola de destino, aplique la protección de AMS a los mensajes de entrada antes de colocarlos en la cola de destino.

Este valor sólo es relevante para canales con un *ChannelType* de MQCHT_RECEIVER o MQCHT_REQUESTER.

Es un campo de entrada para la salida. Este campo no está presente si *Version* es menor que MQCD_VERSION_12.

SSLCipherSpec (MQCHAR32)
Este campo especifica la especificación de cifrado que se utiliza cuando se utiliza TLS.

Si SSLCipherSpec está en blanco, el canal no utiliza TLS. Si no está en blanco, este campo contiene una serie que especifica la CipherSpec en uso.

Este parámetro es válido para todos los tipos de canal. Está soportado en las plataformas siguientes:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Solaris Solaris
- Windows Windows

- z/OS z/OS

Sólo es válido para tipos de canal de un tipo de transporte (TRPTYPE) de TCP.

Es un campo de entrada para la salida. La longitud de este campo la proporciona MQ_SSL_CIPHER_SPEC_LENGTH. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLClientAuth (MQLONG)
Este campo especifica si es necesaria la autenticación de cliente TLS.

Este campo sólo es relevante para las definiciones de canal SVRCONN.

Es uno de los valores siguientes:

MQSCA_REQUIRED
Se requiere autenticación de cliente.

MQSCA_OPTIONAL
Autenticación de cliente opcional.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNameLongitud (MQLONG)
Este campo especifica la longitud en bytes del nombre de igual TLS al que apunta *SSLPeerNamePtr*.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)
Este campo especifica la dirección del nombre de igual TLS.

Cuando se recibe un certificado durante un reconocimiento TLS satisfactorio, el nombre distinguido del asunto del certificado se copia en el campo MQCD al que accede SSLPeerNamePtr al final del canal que recibe el certificado. Sobrescribe el valor SSLPeerName para el canal si este valor está presente en la definición de canal del usuario local. Si se especifica una salida de seguridad en este extremo del canal, recibe el nombre distinguido del certificado de igual en el MQCD.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

Nota: Es posible que sea necesario actualizar las aplicaciones de salida de seguridad construidas antes del release de IBM WebSphere MQ 7.1 . Para obtener más información, consulte Programas de salida de seguridad de canal.

StrucLength (MQLONG)
Este campo especifica la longitud en bytes de la estructura MQCD.

La longitud no incluye ninguna de las series a las que se dirigen los campos de puntero contenidos en la estructura. El valor puede ser uno de los siguientes:

MQCD_LENGTH_4
Longitud de la estructura de definición de canal version-4 .

MQCD_LENGTH_5
Longitud de la estructura de definición de canal version-5 .

MQCD_LENGTH_6
Longitud de la estructura de definición de canal version-6 .

MQCD_LENGTH_7
Longitud de la estructura de definición de canal version-7 .

MQCD_LENGTH_8
Longitud de la estructura de definición de canal version-8 .

MQCD_LENGTH_9
Longitud de la estructura de definición de canal version-9 .

MQCD_LENGTH_10
Longitud de la estructura de definición de canal version-10 .

MQCD_LENGTH_11
Longitud de la estructura de definición de canal version-11 .

z/OS V 9.1.3 **MQCD_LENGTH_12**
Longitud de la estructura de definición de canal version-12 .

La constante siguiente especifica la longitud de la versión actual:

MQCD_LONGITUD_ACTUAL
Longitud de la versión actual de la estructura de definición de canal.

Nota: Estas constantes tienen valores que son específicos del entorno.

El campo no está presente si *Version* es menor que MQCD_VERSION_4.

TpName (MQCHAR64)
Este campo especifica el nombre de programa de transacción de LU 6.2 .

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco en las plataformas en las que la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_TP_NAME_LENGTH.

TransportType (MQLONG)
Este campo especifica el protocolo de transmisión que se va a utilizar.

El valor no se comprueba si el canal se ha iniciado desde el otro extremo.

Es uno de los valores siguientes:

MQXPT_LU62
Protocolo de transporte LU 6.2 .

MQXPT_TCP
Protocolo de transporte TCP/IP.

MQXPT_NETBIOS
Protocolo de transporte NetBIOS .

Este valor está soportado en los entornos siguientes: Windows.

MQXPT_SPX
Protocolo de transporte SPX.

Este valor está soportado en los entornos siguientes: Windows, más los clientes IBM MQ conectados a estos sistemas.

UseDLQ (MQLONG)
Este campo especifica si se utiliza la cola de mensajes no entregados (o la cola de mensajes no entregados) cuando los canales no pueden entregar los mensajes.

Puede contener uno de los valores siguientes:

MQUSEDLQ_NO
Los mensajes que un canal no puede entregar se tratan como un error. El canal descarta el mensaje o el canal finaliza, de acuerdo con el valor NPMSPEED.

MQUSEDLQ_SÍ
Cuando el atributo de gestor de colas DEADQ proporciona el nombre de una cola de mensajes no entregados, se utiliza, de lo contrario, el comportamiento es como para NO. YES es el valor predeterminado.

UserIdentifier (MQCHAR12)
Este campo especifica el identificador de usuario utilizado por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en UNIX y Windows, y sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_2.

Versión (MQLONG)
El campo `Version` especifica el número de versión más alto que puede establecer para la estructura.

El valor depende del entorno:

MQCD _VERSION_1
Estructura de definición de canal de la versión 1.

MQCD _VERSION_2
Estructura de definición de canal de la versión 2.

MQCD _VERSION_3
Estructura de definición de canal de la versión 3.

MQCD _VERSION_4
Estructura de definición de canal de la versión 4.

MQCD _VERSION_5
Estructura de definición de canal de la versión 5.

MQCD _VERSION_6
Estructura de definición de canal de la versión 6.

MQCD _VERSION_7
Estructura de definición de canal de la versión 7.

MQCD _VERSION_8
Estructura de definición de canal de la versión 8.

MQCD _VERSION_9
Estructura de definición de canal de la versión 9.

La versión 9 es la más alta en la que puede establecer el campo en IBM WebSphere MQ 7.0 y IBM WebSphere MQ 7.0.1 en todas las plataformas.

MQCD _VERSION_10
Estructura de definición de canal de la versión 10.

La versión 10 es la más alta en la que puede establecer el campo en IBM WebSphere MQ 7.1 y IBM WebSphere MQ 7.5 en todas las plataformas.

MQCD _VERSION_11
Estructura de definición de canal de la versión 11.

La versión 11 es la más alta en la que puede establecer el campo en IBM MQ 8.0 en todas las plataformas.

z/OS V 9.1.3 **MQCD _VERSION_12**
Estructura de definición de canal de la versión 12.

La versión 12 es la más alta en la que puede establecer el campo en IBM MQ 9.1.3.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCD_CURRENT_VERSION

El valor establecido en MQCD_CURRENT_VERSION es la versión actual de la estructura de definición de canal que se está utilizando.

El valor de MQCD_CURRENT_VERSION depende del entorno. Contiene el valor más alto soportado por la plataforma.

MQCD_CURRENT_VERSION no se utiliza para inicializar las estructuras predeterminadas proporcionadas en los archivos de cabecera, copia e inclusión proporcionados para distintos lenguajes de programación. La inicialización predeterminada de Version depende de la plataforma y del release.

Para IBM WebSphere MQ 7.0 y posteriores, las declaraciones MQCD en los archivos de cabecera, copia e inclusión se inicializan en MQCD_VERSION_6. Para utilizar campos MQCD adicionales, las aplicaciones deben establecer el número de versión en MQCD_CURRENT_VERSION. Si está escribiendo una aplicación que es portable entre varios entornos, debe elegir una versión que esté soportada en todos los entornos.

Consejo: Cuando se introduce una nueva versión de la estructura MQCD , el diseño de la parte existente no cambia. La salida debe comprobar el número de versión. Debe ser igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

XmitQName (MQCHAR48)
Este campo especifica el nombre de la cola de transmisión de la que se recuperan los mensajes.

Este campo sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER o MQCHT_SERVER.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH.

Declaración C

Esta declaración es la declaración C para la estructura MQCD.

V 9.1.3

```
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 */
};
```

declaración COBOL

Esta declaración es la declaración COBOL para la estructura MQCD.

```
V 9.1.3
** 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 **
```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura MQCD.

```
D* MQCD Structure
     D*
     D* Channel definition name
     D  CDCHN                  1     20
     D* Structure version number
     D  CDVER                 21     24I 0
     D* Channel type
     D  CDCHT                 25     28I 0
     D* Transport type
     D  CDTRT                 29     32I 0
     D* Channel description
     D  CDDES                 33     96
     D* Queue manager name
     D  CDQM                  97    144
     D* Transmission queue name
     D  CDXQ                 145    192
     D* First 20 bytes of connection name
     D  CDSCN                193    212
     D* Reserved
     D  CDMCA                213    232
     D* LU 6.2 Mode name
     D  CDMOD                233    240
     D* LU 6.2 transaction program name
     D  CDTP                 241    304
     D* Batch size
     D  CDBS                 305    308I 0
     D* Disconnect interval
     D  CDDI                 309    312I 0
     D* Short retry count
     D  CDSRC                313    316I 0
     D* Short retry wait interval
     D  CDSRI                317    320I 0
     D* Long retry count
     D  CDLRC                321    324I 0
     D* Long retry wait interval
     D  CDLRI                325    328I 0
     D* Channel security exit name
     D  CDSCX                329    348
     D* Channel message exit name
     D  CDMSX                349    368
     D* Channel send exit name
     D  CDSNX                369    388
     D* Channel receive exit name
     D  CDRCX                389    408
     D* Highest allowable message sequence number
     D  CDSNW                409    412I 0
     D* Maximum message length
     D  CDMML                413    416I 0
     D* Put authority
     D  CDPA                 417    420I 0
     D* Data conversion
     D  CDDC                 421    424I 0
     D* Channel security exit user data
     D  CDSCD                425    456
     D* Channel message exit user data
     D  CDMSD                457    488
     D* Channel send exit user data
     D  CDSND                489    520
     D* Channel receive exit user data
     D  CDRCD                521    552
     D* Ver:1 **
     D* User identifier
     D  CDUID                553    564
     D* Password
     D  CDPW                 565    576
     D* First 12 bytes of MCA user identifier
     D  CDAUI                577    588
     D* Message channel agent type
     D  CDCAT                589    592I 0
     D* Connection name
     D  CDCON                593    848
     D  CDCN2                849    856
     D* First 12 bytes of user identifier from partner
     D  CDRUI                857    868
     D* Password from partner
     D  CDRPW                869    880
     D* Ver:2 **
     D* Channel message retry exit name
     D  CDMRX                881    900
     D* Channel message retry exit user data
```

```
     D  CDMRD                 901    932
     D* Number of times MCA will try to put the message, after first
     D* attempt has failed
     D  CDMRC                 933    936I 0
     D* Minimum interval in milliseconds after which the open or put
     D* operation will be retried
     D  CDMRI                 937    940I 0
     D* Ver:3 **
     D* Time in seconds between heartbeat flows
     D  CDHBI                 941    944I 0
     D* Batch duration
     D  CDBI                  945    948I 0
     D* Speed at which nonpersistent messages are sent
     D  CDNPM                 949    952I 0
     D* Length of MQCD structure
     D  CDLEN                 953    956I 0
     D* Length of exit name
     D  CDXNL                 957    960I 0
     D* Length of exit user data
     D  CDXDL                 961    964I 0
     D* Number of message exits defined
     D  CDMXD                 965    968I 0
     D* Number of send exits defined
     D  CDSXD                 969    972I 0
     D* Number of receive exits defined
     D  CDRXD                 973    976I 0
     D* Address of first MsgExit field
     D  CDMXP                 977    992*
     D* Address of first MsgUserData field
     D  CDMUP                 993   1008*
     D* Address of first SendExit field
     D  CDSXP                1009   1024*
     D* Address of first SendUserData field
     D  CDSUP                1025   1040*
     D* Address of first ReceiveExit field
     D  CDRXP                1041   1056*
     D* Address of first ReceiveUserData field
     D  CDRUP                1057   1072*
     D* Ver:4 **
     D* Address of a list of cluster names
     D  CDCLP                1073   1088*
     D* Number of clusters to which the channel belongs
     D  CDCLD                1089   1092I 0
     D* Network priority
     D  CDNP                 1093   1096I 0
     D* Ver:5 **
     D* Length of long MCA user identifier
     D  CDLML                1097   1100I 0
     D* Length of long remote user identifier
     D  CDLRL                1101   1104I 0
     D* Address of long MCA user identifier
     D  CDLMP                1105   1120*
     D* Address of long remote user identifier
     D  CDLRP                1121   1136*
     D* MCA security identifier
     D  CDMSI                1137   1176
     D* Remote security identifier
     D  CDRSI                1177   1216
     D* Ver:6 **
     D* 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 **
```

Declaración de ensamblador System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura MQCD.

```
V 9.1.3
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)
```

Declaración de Visual Basic

Esta declaración es la declaración de Visual Basic de la estructura MQCD.

En Visual Basic, la estructura MQCD se puede utilizar con la estructura MQCNO en la llamada MQCONNX.

```
Type MQCD
  ChannelName            As String*20  'Channel definition name'
  Version                As Long       'Structure version number'
  ChannelType            As Long       'Channel type'
  TransportType          As Long       'Transport type'
  Desc                   As String*64  'Channel description'
  QMgrName               As String*48  'Queue manager name'
  XmitQName              As String*48  'Transmission queue name'
  ShortConnectionName    As String*20  'First 20 bytes of connection'
                                       'name'
  MCAName                As String*20  'Reserved'
  ModeName               As String*8   'LU 6.2 Mode name'
  TpName                 As String*64  'LU 6.2 transaction program name'
  BatchSize              As Long       'Batch size'
  DiscInterval           As Long       'Disconnect interval'
  ShortRetryCount        As Long       'Short retry count'
  ShortRetryInterval     As Long       'Short retry wait interval'
  LongRetryCount         As Long       'Long retry count'
  LongRetryInterval      As Long       'Long retry wait interval'
  SecurityExit           As String*128 'Channel security exit name'
  MsgExit                As String*128 'Channel message exit name'
  SendExit               As String*128 'Channel send exit name'
  ReceiveExit            As String*128 'Channel receive exit name'
  SeqNumberWrap          As Long       'Highest allowable message'
                                       'sequence number'
  MaxMsgLength           As Long       'Maximum message length'
  PutAuthority           As Long       'Put authority'
  DataConversion         As Long       'Data conversion'
  SecurityUserData       As String*32  'Channel security exit user data'
  MsgUserData            As String*32  'Channel message exit user data'
  SendUserData           As String*32  'Channel send exit user data'
  ReceiveUserData        As String*32  'Channel receive exit user data'
  UserIdentifier         As String*12  'User identifier'
  Password               As String*12  'Password'
  MCAUserIdentifier      As String*12  'First 12 bytes of MCA user'
                                       'identifier'
  MCAType                As Long       'Message channel agent type'
  ConnectionName         As String*264 'Connection name'
  RemoteUserIdentifier   As String*12  'First 12 bytes of user'
                                       'identifier from partner'
  RemotePassword         As String*12  'Password from partner'
  MsgRetryExit           As String*128 'Channel message retry exit name'
  MsgRetryUserData       As String*32  'Channel message retry exit user'
                                       'data'
  MsgRetryCount          As Long       'Number of times MCA will try to'
                                       'put the message, after the'
                                       'first attempt has failed'
  MsgRetryInterval       As Long       'Minimum interval in'
                                       'milliseconds after which the'
                                       'open or put operation will be'
                                       'retried'
  HeartbeatInterval      As Long       'Time in seconds between'
                                       'heartbeat flows'
  BatchInterval          As Long       'Batch duration'
  NonPersistentMsgSpeed  As Long       'Speed at which nonpersistent'
                                       'messages are sent'
  StrucLength            As Long       'Length of MQCD structure'
  ExitNameLength         As Long       'Length of exit name'
  ExitDataLength         As Long       'Length of exit user data'
```

```
  MsgExitsDefined        As Long       'Number of message exits defined'
  SendExitsDefined       As Long       'Number of send exits defined'
  ReceiveExitsDefined    As Long       'Number of receive exits defined'
  MsgExitPtr             As MQPTR      'Address of first MsgExit field'
  MsgUserDataPtr         As MQPTR      'Address of first MsgUserData'
                                       'field'
  SendExitPtr            As MQPTR      'Address of first SendExit field'
  SendUserDataPtr        As MQPTR      'Address of first SendUserData'
                                       'field'
  ReceiveExitPtr         As MQPTR      'Address of first ReceiveExit'
                                       'field'
  ReceiveUserDataPtr     As MQPTR      'Address of first'
                                       'ReceiveUserData field'
  ClusterPtr             As MQPTR      'Address of a list of cluster'
                                       'names'
  ClustersDefined        As Long       'Number of clusters to which the'
                                       'channel belongs'
  NetworkPriority        As Long       'Network priority'
  LongMCAUserIdLength    As Long       'Length of long MCA user'
                                       'identifier'
  LongRemoteUserIdLength As Long       'Length of long remote user'
                                       'identifier'
  LongMCAUserIdPtr       As MQPTR      'Address of long MCA user'
                                       'identifier'
  LongRemoteUserIdPtr    As MQPTR      'Address of long remote user'
                                       'identifier'
  MCASecurityId          As MQBYTE40   'MCA security identifier'
  RemoteSecurityId       As MQBYTE40   'Remote security identifier'
  SSLCipherSpec          As String*32  '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
```

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Si un programa de salida de canal cambia un campo en la estructura de datos MQCD, el nuevo valor normalmente lo ignora el proceso de canal de IBM MQ . Sin embargo, el nuevo valor permanece en el MQCD y se le pasa a las salidas restantes en una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Si SharingConversations se establece en FALSE en la estructura MQCXP, se pueden realizar cambios en determinados campos, en función del tipo de programa de salida, el tipo de canal y el código de razón de salida. La tabla siguiente muestra los campos que se pueden cambiar y afectan al comportamiento del canal, y en qué circunstancias. Si un programa de salida cambia uno de estos campos en cualquier otra circunstancia, o cualquier campo no listado, el nuevo valor es ignorado por el proceso de canal. El nuevo valor permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Cualquier tipo de programa de salida cuando se llama para la inicialización (MQXR_INIT) puede cambiar el campo ChannelName de cualquier tipo de canal, siempre que MQCXP SharingConverstions esté establecido en FALSE. Sólo una salida de seguridad puede cambiar el campo MCAUserIdentifier , independientemente del valor de MQCXP SharingConverstions.

Tabla 823. Campos que se pueden cambiar y que afectan al comportamiento del canal			
Campo	**Código de razón de salida**	**Tipo de salida**	**Tipo de canal**
ChannelName	MQXR_INIT	Todo	Todo
TransportType	MQXR_INIT	Todo	Todo
XmitQName	MQXR_INIT	Todo	SDR, RCVR
ModeName	MQXR_INIT	Todo	Todo
TpName	MQXR_INIT	Todo	Todo
BatchSize	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Recuento de ShortRetry	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
shortRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Recuento de LongRetry	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
longRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SeqNumberAjustar	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgLength	MQXR_INIT	Todo	Todo

Tabla 823. Campos que se pueden cambiar y que afectan al comportamiento del canal (continuación)			
Campo	**Código de razón de salida**	**Tipo de salida**	**Tipo de canal**
PutAuthority	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Todo	Todo
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	RCVR, RQSTR, SVRCONN, CLUSRCVR
ConnectionName	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
Recuento de MsgRetry	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
Intervalo de MsgRetry	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Todo	Todo
BatchInterval	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Todo	Todo
SSLPeerNamePtr	MQXR_INIT	Todo	Todo
SSLPeerNameLongitud	MQXR_INIT	Todo	Todo

Tabla 823. Campos que se pueden cambiar y que afectan al comportamiento del canal (continuación)			
Campo	**Código de razón de salida**	**Tipo de salida**	**Tipo de canal**
SSLClientAuth	MQXR_INIT	Todo	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
Intervalo de mantenimiento de activación	MQXR_INIT	Todo	Todo
LocalAddress	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
Lista de HdrComp	MQXR_INIT	Todo	Todo
Lista de MsgComp	MQXR_INIT	Todo	Todo
ChannelMonitoring	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Todo	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP-Parámetro de salida de canal

La estructura MQCXP se pasa a cada tipo de salida llamada por un agente de canal de mensajes (MCA), un canal de conexión de cliente o un canal de conexión de servidor.

Consulte MQ_CHANNEL_EXIT.

Los campos descritos como "entrada a la salida" en las descripciones que siguen son ignorados por el canal cuando la salida devuelve el control al canal. Los campos de entrada que la salida cambie en el bloque de parámetros de salida de canal no se conservarán para su siguiente invocación. Los cambios realizados en los campos de entrada/salida (por ejemplo, el campo *ExitUserArea*) se conservan sólo para las invocaciones de esa instancia de la salida. Estos cambios no se pueden utilizar para pasar datos entre salidas diferentes definidas en el mismo canal, o entre la misma salida definida en canales diferentes.

Referencia relacionada

"Campos" en la página 1560
Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

"Declaración C" en la página 1571
Esta declaración es la declaración C para la estructura MQCXP.

"declaración COBOL" en la página 1572
Esta declaración es la declaración COBOL para la estructura MQCXP.

"Declaración RPG (ILE)" en la página 1573
Esta declaración es la declaración RPG para la estructura MQCXP.

"Declaración de ensamblador System/390" en la página 1574
Esta declaración es la declaración de ensamblador System/390 para la estructura MQCXP.

Campos

Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

StrucId (MQCHAR4)
Este campo especifica el identificador de estructura.

El valor debe ser:

MQCXP_STRUC_ID
Identificador de la estructura de parámetros de salida de canal.

Para el lenguaje de programación C, también se define la constante MQCXP_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQCXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Versión (MQLONG)
Este campo especifica el número de versión de la estructura.

El valor depende del entorno:

MQCXP_VERSION_1
Estructura del parámetro de salida de canal Version-1 .

MQCXP_VERSION_3
Estructura del parámetro de salida de canal Version-3 .

UNIX El campo tiene este valor en sistemas UNIX no listados en otros lugares.

MQCXP_VERSION_4
Estructura del parámetro de salida de canal Version-4 .

MQCXP_VERSION_5
Estructura del parámetro de salida de canal Version-5 .

MQCXP_VERSION_6
Estructura del parámetro de salida de canal Version-6 .

MQCXP_VERSION_8
Estructura del parámetro de salida de canal Version-8 .

z/OS El campo tiene este valor en z/OS.

MQCXP_VERSION_9
Estructura del parámetro de salida de canal Version-9 .

El campo tiene este valor en los entornos siguientes:

- AIX AIX
- IBM i IBM i

- Linux
- Solaris
- Windows
- z/OS

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCXP_CURRENT_VERSION
Versión actual de la estructura de parámetros de salida de canal.

El valor depende del entorno.

Nota: Cuando se introduce una nueva versión de la estructura MQCXP, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

ExitId (MQLONG)
Este campo especifica el tipo de salida que se llama y se establece en la entrada de la rutina de salida.

Son posibles los siguientes valores:

MQXT_CHANNEL_SEC_EXIT
Salida de seguridad de canal.

MQXT_CHANNEL_MSG_EXIT
Salida de mensajes de canal.

MQXT_CHANNEL_SEND_EXIT
Salida de envío de canal.

MQXT_CHANNEL_RCV_EXIT
Salida de recepción de canal.

MQXT_CHANNEL_MSG_RETRY_EXIT
Salida de reintento de mensaje de canal.

MQXT_CHANNEL_AUTO_DEF_EXIT
Salida de definición automática de canal.

En z/OS, este tipo de salida sólo está soportado para canales de tipo MQCHT_CLUSSDR y MQCHT_CLUSRCVR.

Es un campo de entrada para la salida.

ExitReason (MQLONG)
Este campo especifica la razón por la que se llama a la salida y se establece en la entrada de la rutina de salida.

La salida de definición automática no la utiliza. Son posibles los siguientes valores:

MQXR_INIT
Salga de la inicialización.

Este valor indica que la salida se está invocando por primera vez. Permite que la salida adquiera e inicialice los recursos que necesite (por ejemplo: memoria).

MQXR_TERM
Terminación de salida.

Este valor indica que la salida está a punto de terminar. La salida debe liberar los recursos que haya adquirido desde que se inicializó (por ejemplo: memoria).

MQXR_MSG
Procesar un mensaje.

Este valor indica que se está invocando la salida para procesar un mensaje. Este valor sólo se produce para salidas de mensajes de canal.

MQXR_XMIT
Procesar una transmisión.

Este valor sólo se produce para salidas de envío y recepción de canal.

MQXR_SEC_MSG
Se ha recibido el mensaje de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

MQXR_INIT_SEC
Inicie el intercambio de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

La salida de seguridad del receptor siempre se invoca con esta razón inmediatamente después de ser invocada con MQXR_INIT, para darle la oportunidad de iniciar un intercambio de seguridad. Si rechaza la oportunidad (devolviendo MQXCC_OK en lugar de MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente se invoca con MQXR_INIT_SEC.

Si la salida de seguridad del receptor no inicia un intercambio de seguridad (devolviendo MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente nunca se invoca con MQXR_INIT_SEC; en su lugar, se invoca con MQXR_SEC_MSG para procesar el mensaje del receptor. (En cualquier caso, se invoca por primera vez con MQXR_INIT.)

A menos que una de las salidas de seguridad solicite la terminación del canal (estableciendo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL), el intercambio de seguridad debe completarse en el lado que ha iniciado el intercambio. Por lo tanto, si se invoca una salida de seguridad con MQXR_INIT_SEC y inicia un intercambio, la próxima vez que se invoque la salida será con MQXR_SEC_MSG. Esto sucede si hay un mensaje de seguridad para que la salida se procese o no. Hay un mensaje de seguridad si el socio devuelve MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG, pero no si el socio devuelve MQXCC_OK o no hay ninguna salida de seguridad en el socio. Si no hay ningún mensaje de seguridad para procesar, la salida de seguridad en el extremo de inicio se vuelve a invocar con un *DataLength* de cero.

MQXR_REINTENTAR
Vuelva a intentar un mensaje.

Este valor sólo se produce para salidas de reintento de mensaje.

MQXR_AUTO_CLUSSDR
Definición automática de un canal de clúster emisor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_RECEIVER
Definición automática de un canal receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_SVRCONN
Definición automática de un canal de conexión de servidor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_CLUSRCVR
Definición automática de un canal de clúster receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_SEC_PARMS
Parámetros de seguridad

Este valor sólo se aplica a las salidas de seguridad e indica que se está pasando una estructura MQCSP a la salida. Para obtener más información, consulte "MQCSP-Parámetros de seguridad" en la página 336.

Nota:

1. Si tiene más de una salida definida para un canal, cada uno de ellos se invoca con MQXR_INIT cuando se inicializa el MCA. Además, cada uno de ellos se invoca con MQXR_TERM cuando termina el MCA.
2. Para la salida de definición automática de canal, *ExitReason* no se establece si *Version* es menor que MQCXP_VERSION_4. El valor MQXR_AUTO_SVRCONN está implícito en este caso.

Es un campo de entrada para la salida.

ExitResponse (MQLONG)
Este campo especifica la respuesta de la salida.

Este campo lo establece la salida para comunicarse con el MCA. Tiene que ser uno de los valores siguientes:

MQXCC_Correcto
La salida se ha completado correctamente.

- Para la salida de seguridad de canal, este valor indica que la transferencia de mensajes ahora puede continuar con normalidad.
- Para la salida de reintento de mensaje de canal, este valor indica que el MCA debe esperar el intervalo de tiempo devuelto por la salida en el campo *MsgRetryInterval* en MQCXP y, a continuación, vuelva a intentar el mensaje.

El campo *ExitResponse2* puede contener información adicional.

MQXCC_SUPPRESS_FUNCTION
Suprimir función.

- Para la salida de seguridad de canal, este valor indica que el canal debe terminar.
- Para la salida de mensajes de canal, este valor indica que el mensaje no debe continuar hacia su destino. En su lugar, el MCA genera un mensaje de informe de excepción (si lo ha solicitado el remitente del mensaje original) y coloca el mensaje contenido en el almacenamiento intermedio original en la cola de mensajes no entregados (si el remitente ha especificado MQRO_DEAD_LETTER_Q), o lo descarta (si el remitente ha especificado MQRO_DISCARD_MSG).

  Para los mensajes persistentes, si el remitente ha especificado MQRO_DEAD_LETTER_Q, pero la colocación en la cola de mensajes no entregados falla, o no hay ninguna cola de mensajes no entregados, el mensaje original se deja en la cola de transmisión y el mensaje de informe no se genera. El mensaje original también se deja en la cola de transmisión si el mensaje de informe no se puede generar correctamente.

  El campo *Feedback* de la estructura MQDLH al principio del mensaje en la cola de mensajes no entregados indica por qué el mensaje se ha colocado en la cola de mensajes no entregados; este código de retorno también se utiliza en el descriptor de mensaje del mensaje de informe de excepción (si el remitente ha solicitado uno).
- Para la salida de reintento de mensaje de canal, este valor indica que el MCA no espera y vuelve a intentar el mensaje; en su lugar, el MCA continúa inmediatamente con su proceso de anomalía normal (el mensaje se coloca en la cola de mensajes no entregados o se descarta, tal como especifica el emisor del mensaje).
- Para la salida de definición automática de canal, debe especificarse MQXCC_OK o MQXCC_SUPPRESS_FUNCTION. Si no se especifica ninguno de estos valores, se presupone MQXCC_SUPPRESS_FUNCTION de forma predeterminada y se abandona la definición automática.

Esta respuesta no está soportada para las salidas de envío y recepción de canal.

MQXCC_SEND_SEC_MSG
Enviar mensaje de seguridad.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica que la salida ha proporcionado un mensaje de seguridad que debe transmitirse al socio.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Envíe un mensaje de seguridad que requiera una respuesta.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica

- que la salida ha proporcionado un mensaje de seguridad que puede transmitirse al socio, y
- que la salida requiere una respuesta del socio. Si no se recibe ninguna respuesta, el canal debe terminar, porque la salida todavía no ha decidido si las comunicaciones pueden continuar.

MQXCC_SUPPRESS_EXIT
Suprimir salida.

- Este valor lo pueden establecer todos los tipos de salida de canal que no sean una salida de seguridad o una salida de definición automática. Suprime cualquier invocación adicional de esa salida (como si su nombre hubiera estado en blanco en la definición de canal), hasta la terminación del canal, cuando se vuelve a invocar la salida con un *ExitReason* de MQXR_TERM.
- Si una salida de reintento de mensaje devuelve este valor, los reintentos de mensaje para los mensajes posteriores se controlan mediante los atributos de canal *MsgRetryCount* y *MsgRetryInterval* como normales. Para el mensaje actual, el MCA realiza el número de reintentos pendientes, a intervalos proporcionados por el atributo de canal *MsgRetryInterval* , pero sólo si el código de razón es uno que el MCA reintentaría normalmente (consulte el campo *MsgRetryCount* descrito en "MQCD-Definición de canal" en la página 1517). El número de reintentos pendientes es el valor del atributo **MsgRetryCount** , menos el número de veces que la salida ha devuelto MQXCC_OK para el mensaje actual; si este número es negativo, el MCA no realiza más reintentos para el mensaje actual.

MQXCC_CLOSE_CHANNEL
Cierre el canal.

Este valor lo puede establecer cualquier tipo de salida de canal excepto una salida de definición automática.

Si la compartición de conversaciones no está habilitada, este valor cierra el canal.

Si la compartición de conversaciones está habilitada, este valor finaliza la conversación. Si esta conversación es la única conversación en el canal, el canal también se cierra.

Este campo es un campo de entrada/salida de la salida.

ExitResponse2 (MQLONG)
Este campo especifica la respuesta secundaria de la salida.

Este campo se establece en cero en la entrada a la rutina de salida. Se puede establecer mediante la salida para proporcionar más información a las funciones de canal de IBM MQ . La salida de definición automática no la utiliza.

La salida puede establecer uno o varios de los valores siguientes. Si se necesita más de uno, se añaden los valores. Las combinaciones que no son válidas se anotan; se permiten otras combinaciones.

MQXR2_PUT_WITH_DEF_ACTION
Poner con acción predeterminada.

Este valor lo establece la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con la acción predeterminada del MCA, que es el ID de usuario predeterminado del MCA, o el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_PUT_WITH_DEF_USERID
Colocar con identificador de usuario predeterminado.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el identificador de usuario predeterminado del MCA.

MQXR2_PUT_WITH_MSG_USERID
Transferir con el identificador de usuario del mensaje.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje (esto puede haber sido modificado por la salida).

Solo debe establecerse uno de MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERIDy MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER
Utilice el almacenamiento intermedio del agente.

Este valor indica que los datos que deben pasarse están en *AgentBuffer*, no en *ExitBufferAddr*.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_USE_EXIT_BUFFER
Utilice el almacenamiento intermedio de salida.

Este valor indica que los datos que deben pasarse están en *ExitBufferAddr*, no en *AgentBuffer*.

Solo se debe establecer uno de MQXR2_USE_AGENT_BUFFER y MQXR2_USE_EXIT_BUFFER .

MQXR2_DEFAULT_CONTINUATION
Continuación predeterminada.

La continuación con la siguiente salida de la cadena depende de la respuesta de la última salida invocada:

- Si se devuelve MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL, no se llama a más salidas de la cadena.
- De lo contrario, se invoca la siguiente salida de la cadena.

MQXR2_CONTINUE_CHAIN
Continúe con la siguiente salida.

MQXR2_SUPPRESS_CHAIN
Omita las salidas restantes de la cadena.

Es un campo de entrada/salida para la salida.

Feedback (MQLONG)
Este campo especifica el código de comentarios.

Este campo se establece en MQFB_NONE en la entrada a la rutina de salida.

Si una salida de mensajes de canal establece el campo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION, el campo *Feedback* especifica el código de retorno que identifica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message) y también se utiliza para enviar un informe de excepción si se ha solicitado uno. En este caso, si el campo *Feedback* es MQFB_NONE, se utiliza el siguiente código de retorno:

MQFB_STOPPED_BY_MSG_EXIT
Mensaje detenido por salida de mensajes de canal.

El MCA no utiliza el valor devuelto en este campo por las salidas de seguridad de canal, envío, recepción y reintento de mensaje.

El valor devuelto en este campo por las salidas de definición automática no se utiliza si *ExitResponse* es MQXCC_OK, pero de lo contrario se utiliza para el parámetro *AuxErrorDataInt1* en el mensaje de suceso.

Es un campo de entrada/salida de la salida.

MaxSegmentLongitud (MQLONG)
Este campo especifica la longitud máxima en bytes que se puede enviar en una sola transmisión.

La salida de definición automática no la utiliza. Es de interés para una salida de emisión de canal, porque esta salida debe asegurarse de que no aumenta el tamaño de un segmento de transmisión a un valor mayor que *MaxSegmentLength*. La longitud incluye los 8 bytes iniciales que la salida no debe cambiar. El valor se negocia entre las funciones de canal de IBM MQ cuando se inicia el canal. Consulte Escritura de programas de salida de canal para obtener más información sobre las longitudes de segmento.

El valor de este campo no es significativo si *ExitReason* es MQXR_INIT.

Es un campo de entrada para la salida.

Área ExitUser(MQBYTE16)
Este campo especifica el área de usuario de salida: un campo disponible para que lo utilice la salida.

Se inicializa a cero binario antes de la primera invocación de la salida (que tiene un *ExitReason* establecido en MQXR_INIT) y, a partir de entonces, los cambios realizados en este campo por la salida se conservan entre las invocaciones de la salida.

Se define el valor siguiente:

MQXUA_NONE
No hay información de usuario.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY; esta constante tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH. Es un campo de entrada/salida para la salida.

ExitData (MQCHAR32)
Este campo especifica los datos de salida.

Este campo se establece en la entrada de la rutina de salida en la información que las funciones de canal de IBM MQ han tomado de la definición de canal. Si no hay dicha información disponible, este campo está en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_2.

Recuento de MsgRetry(MQLONG)
Este campo especifica el número de veces que se ha reintentado el mensaje.

La primera vez que se invoca la salida para un mensaje determinado, este campo tiene el valor cero (todavía no se han intentado reintentos). En cada invocación posterior de la salida para ese mensaje, el valor se incrementa en uno por el MCA.

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Intervalo MsgRetry(MQLONG)
Este campo especifica el intervalo mínimo en milisegundos después del cual se reintenta la operación de transferencia.

La primera vez que se invoca la salida para un mensaje determinado, este campo contiene el valor del atributo de canal *MsgRetryInterval* . La salida puede dejar el valor sin modificar o modificarlo para especificar un intervalo de tiempo diferente en milisegundos. Si la salida devuelve MQXCC_OK en *ExitResponse*, el MCA espera al menos este intervalo de tiempo antes de reintentar la operación MQOPEN o MQPUT. El intervalo de tiempo especificado debe ser cero o mayor.

La segunda y siguientes veces que se invoca la salida para ese mensaje, este campo contiene el valor devuelto por la invocación anterior de la salida.

Si el valor devuelto en el campo *MsgRetryInterval* es menor que cero o mayor que 999 999 999 999, y *ExitResponse* es MQXCC_OK, el MCA ignora el campo *MsgRetryInterval* en MQCXP y espera en su lugar el intervalo especificado por el atributo de canal *MsgRetryInterval* .

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Razón MsgRetry(MQLONG)
Este campo especifica el código de razón del intento anterior de colocar el mensaje.

Este campo es el código de razón del intento anterior de colocar el mensaje; es uno de los valores MQRC_*.

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_3.

HeaderLength (MQLONG)
Este campo especifica la longitud de la información de cabecera.

Este campo sólo es relevante para una salida de mensajes y una salida de reintento de mensajes. El valor es la longitud de las estructuras de cabecera de direccionamiento al principio de los datos del mensaje; son la estructura MQXQH, la MQMDE (cabecera de extensión de descripción de mensaje) y (para un mensaje de lista de distribución) la estructura MQDH y las matrices de registros MQOR y MQPMR que siguen a la estructura MQXQH.

La salida de mensajes puede examinar esta información de cabecera y modificarla si es necesario, pero los datos que devuelve la salida deben estar en el formato correcto. La salida no debe, por ejemplo, cifrar o comprimir los datos de cabecera en el extremo emisor, incluso si la salida de mensaje en el extremo receptor realiza cambios compensatorios.

Si la salida de mensajes modifica la información de cabecera de forma que cambie su longitud (por ejemplo, añadiendo otro destino a un mensaje de lista de distribución), debe cambiar el valor de *HeaderLength* correspondiente antes de devolver.

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

PartnerName (MQCHAR48)
Este campo especifica el nombre del socio.

El nombre del socio, como se indica a continuación:

- Para canales SVRCONN, es el ID de usuario conectado en el cliente.
- Para todos los demás tipos de canal, es el nombre del gestor de colas del socio.

Cuando se inicializa la salida, este campo está en blanco porque el gestor de colas no conoce el nombre del socio hasta después de que se haya realizado la negociación inicial.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

FAPLevel (MQLONG)
Nivel de protocolos y formatos negociados.

Es un campo de entrada para la salida. Los cambios en este campo sólo se deben realizar bajo la dirección del servicio de IBM . El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

CapabilityFlags (MQLONG)
Puede establecer el distintivo de prestación en MQCF_NONE o MQCF_DIST_LISTS.

Puede establecer cualquiera de los siguientes distintivos de prestación:

MQCF_NONE
Sin distintivos.

MQCF_DIST_LISTS
Listas de distribución soportadas.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

ExitNumber (MQLONG)
Este campo especifica el número ordinal de la salida.

El número ordinal de la salida, dentro del tipo definido en *ExitId*. Por ejemplo, si la salida que se invoca es la tercera salida de mensajes definida, este campo contiene el valor 3. Si el tipo de salida es uno para el que no se puede definir una lista de salidas (por ejemplo, una salida de seguridad), este campo tiene el valor 1.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_5.

ExitSpace (MQLONG)
Este campo especifica el número de bytes en el almacenamiento intermedio de transmisión reservado para que lo utilice la salida.

Este campo sólo es relevante para una salida de envío. Especifica la cantidad de espacio en bytes que las funciones de canal de IBM MQ reservan en el almacenamiento intermedio de transmisión para que la utilice la salida. Este campo permite que la salida añada al almacenamiento intermedio de transmisión una pequeña cantidad de datos (normalmente no más de unos pocos cientos de bytes) para que los utilice una salida de recepción complementaria en el otro extremo. La salida de recepción debe eliminar los datos añadidos por la salida de envío.

El valor siempre es cero en z/OS.

Nota: Este recurso no se debe utilizar para enviar grandes cantidades de datos, ya que puede degradar el rendimiento o incluso inhibir el funcionamiento del canal.

Al establecer *ExitSpace* la salida se garantiza que siempre hay al menos ese número de bytes disponibles en el almacenamiento intermedio de transmisión para que los utilice la salida. Sin embargo, la salida puede utilizar menos de la cantidad reservada, o más de la cantidad reservada si hay espacio disponible en el almacenamiento intermedio de transmisión. El espacio de salida en el almacenamiento intermedio se proporciona siguiendo los datos existentes.

La salida sólo puede establecer *ExitSpace* cuando *ExitReason* tiene el valor MQXR_INIT; en todos los demás casos, el valor devuelto por la salida se ignora. En la entrada a la salida, *ExitSpace* es cero para la llamada MQXR_INIT y es el valor devuelto por la llamada MQXR_INIT en otros casos.

Si el valor devuelto por la llamada MQXR_INIT es negativo, o si hay menos de 1024 bytes disponibles en el almacenamiento intermedio de transmisión para los datos de mensaje después de reservar el espacio

de salida solicitado para todas las salidas de envío de la cadena, el MCA genera un mensaje de error y cierra el canal. De forma similar, si durante la transferencia de datos las salidas de la cadena de salida de envío asignan más espacio de usuario del que reservaron, de manera que quedan menos de 1024 bytes en el almacenamiento intermedio de transmisión para los datos de mensaje, el MCA genera un mensaje de error y cierra el canal. El límite de 1024 permite que los flujos de control y administrativos del canal sean procesados por la cadena de salidas de envío, sin necesidad de que los flujos sean segmentados.

Es un campo de entrada/salida para la salida si *ExitReason* es MQXR_INIT, y un campo de entrada en todos los demás casos. El campo no está presente si *Version* es menor que MQCXP_VERSION_5.

SSLCertUserId (MQCHAR12)
Este campo especifica el UserId asociado con el certificado remoto.

Está en blanco en todas las plataformas excepto z/OS

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

SSLRemCertIssNameLongitud (MQLONG)
Este campo especifica la longitud en bytes del nombre distinguido completo del emisor del certificado remoto al que apunta SSLCertRemoteIssuerNamePtr.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6. El valor es cero si no es un canal TLS.

SSLRemCertIssNamePtr (PMQVOID)
Este campo especifica la dirección del nombre distinguido completo del emisor del certificado remoto.

Su valor es el puntero nulo si no es un canal TLS.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Nota: El comportamiento de las salidas de seguridad de canal al determinar el nombre distinguido del sujeto y el nombre distinguido del emisor se cambia de IBM WebSphere MQ 7.1. Para obtener más información, consulte Programas de salida de seguridad de canal.

SecurityParms (PMQCSP)
Este campo especifica la dirección de la estructura MQCSP utilizada para especificar un ID de usuario y una contraseña.

El valor inicial de este campo es el puntero nulo.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

El valor de este campo devuelto por la salida debe ser utilizable por IBM MQ hasta MQXR_TERM.

CurHdrCompresión (MQLONG)
Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos de cabecera.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA
Se lleva a cabo la compresión de datos de cabecera.

El valor se puede modificar mediante la salida de mensajes de un canal emisor a uno de los valores soportados negociados a los que se accede desde el campo de lista HdrCompde la MQCD. Esto permite que la técnica utilizada para comprimir los datos de cabecera se elija para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Compresión CurMsg(MQLONG)
Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos del mensaje.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_RLE
Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

El valor se puede modificar mediante la salida de mensajes de un canal emisor a uno de los valores soportados negociados a los que se accede desde el campo de lista MsgCompde la MQCD. Esto permite que la técnica utilizada para comprimir los datos del mensaje se decida para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Hconn (MQHCONN)
Este campo especifica el descriptor de conexión que utiliza la salida si necesita realizar alguna llamada MQI dentro de la salida.

Este campo no es relevante para las salidas que se ejecutan en canales de conexión de cliente, donde contiene el valor MQHC_UNUSABLE_HCONN (-1).

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

SharingConversations (MQBOOL)
Este campo especifica si la conversación es la única que se puede ejecutar actualmente en esta instancia de canal, o si actualmente se puede ejecutar más de una conversación en esta instancia de canal.

También indica si el programa de salida está sujeto al riesgo de que otro programa de salida que se ejecuta al mismo tiempo altere el MQCD.

Este campo sólo es relevante para los programas de salida que se ejecutan en canales de conexión de cliente o de conexión de servidor.

Se establece en uno de los valores siguientes:

FALSE
La instancia de salida es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. Esto permite a la salida actualizar de forma segura los campos MQCD sin contención de otras salidas que se ejecutan en otras instancias de canal. Si el canal actúa sobre los cambios en los campos MQCD se define mediante la tabla de campos MQCD en "Cambio de campos MQCD en una salida de canal" en la página 1556.

TRUE
La instancia de salida no es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. El canal no actúa sobre los cambios realizados en MQCD, excepto los cambios listados en la tabla de campos MQCD en "Cambio de campos MQCD en una salida de canal" en la página 1556 por razones de salida que no sean MQXR_INIT. Si esta salida actualiza los

campos MQCD, asegúrese de que no haya ninguna contención de otras salidas en ejecución en otras conversaciones al mismo tiempo proporcionando serialización entre las salidas que se ejecutan en esta instancia de canal.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

MCAUserSource (MQLONG)
Este campo especifica el origen del ID de usuario de MCA proporcionado.

Puede contener uno de los valores siguientes:

MQUSRC_MAP
El ID de usuario se especifica en el atributo MCAUSER.

MQUSRC_CHANNEL
El ID de usuario fluye desde el socio de entrada o se especifica en el campo MCAUSER definido en el objeto de canal.

Es un campo de entrada para la salida. El campo no está presente si la versión es menor que MQCXP_VERSION_8.

Puntos pEntry(PMQIEP)
Este campo especifica la dirección del punto de entrada de interfaz para la llamada MQI o DCI.

El campo no está presente si *Versión* es menor que MQCXP_VERSION_8.

RemoteProduct (MQCHAR4)
Este campo especifica el nombre del producto remoto.

Este campo identifica el producto remoto del cliente, por ejemplo, C o Java, tal como se muestra en el campo **RPRODUCT** de DISPLAY CHSATUS.

El campo no está presente si *Versión* es menor que MQCXP_VERSION_9.

RemoteVersion (MQCHAR8)
Este campo especifica el nombre de la versión remota.

Este campo identifica la versión de las bibliotecas de cliente, tal como se muestra en el campo **RVERSION** de DISPLAY CHSTATUS.

El campo no está presente si *Versión* es menor que MQCXP_VERSION_9.

Declaración C

Esta declaración es la declaración C para la estructura MQCXP.

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Type of exit */
  MQLONG    ExitReason;        /* Reason for invoking exit */
  MQLONG    ExitResponse;      /* Response from exit */
  MQLONG    ExitResponse2;     /* Secondary response from exit */
  MQLONG    Feedback;          /* Feedback code */
  MQLONG    MaxSegmentLength;  /* Maximum segment length */
  MQBYTE16  ExitUserArea;      /* Exit user area */
  MQCHAR32  ExitData;          /* Exit data */
  MQLONG    MsgRetryCount;     /* Number of times the message has been
                                  retried */
  MQLONG    MsgRetryInterval;  /* Minimum interval in milliseconds after
                                  which the put operation should be
                                  retried */
  MQLONG    MsgRetryReason;    /* Reason code from previous attempt to
                                  put the message */
  MQLONG    HeaderLength;      /* Length of header information */
  MQCHAR48  PartnerName;       /* Partner Name */
  MQLONG    FAPLevel;          /* Negotiated Formats and Protocols
                                  level */
  MQLONG    CapabilityFlags;   /* Capability flags */
```

```
  MQLONG    ExitNumber;        /* Exit number */
  /* Ver:3 */
  /* Ver:4 */
  MQLONG    ExitSpace;         /* Number of bytes in transmission buffer
                                  reserved for exit to use */
  /* Ver:5 */
  MQCHAR12  SSLCertUserid;     /* User identifier associated
                                  with remote 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 */
};
```

declaración COBOL

Esta declaración es la declaración COBOL para la estructura MQCXP.

```
**   MQCXP structure
  10 MQCXP.
**    Structure identifier
   15 MQCXP-STRUCID          PIC X(4).
**    Structure version number
   15 MQCXP-VERSION          PIC S9(9) BINARY.
**    Type of exit
   15 MQCXP-EXITID           PIC S9(9) BINARY.
**    Reason for invoking exit
   15 MQCXP-EXITREASON       PIC S9(9) BINARY.
**    Response from exit
   15 MQCXP-EXITRESPONSE     PIC S9(9) BINARY.
**    Secondary response from exit
   15 MQCXP-EXITRESPONSE2    PIC S9(9) BINARY.
**    Feedback code
   15 MQCXP-FEEDBACK         PIC S9(9) BINARY.
**    Maximum segment length
   15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
**    Exit user area
   15 MQCXP-EXITUSERAREA     PIC X(16).
**    Exit data
   15 MQCXP-EXITDATA         PIC X(32).
**    Number of times the message has been retried
   15 MQCXP-MSGRETRYCOUNT    PIC S9(9) BINARY.
**    Minimum interval in milliseconds after which the put operation
**    should be retried
   15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
**    Reason code from previous attempt to put the message
   15 MQCXP-MSGRETRYREASON   PIC S9(9) BINARY.
**    Length of header information
   15 MQCXP-HEADERLENGTH     PIC S9(9) BINARY.
**    Partner Name
   15 MQCXP-PARTNERNAME      PIC X(48).
**    Negotiated Formats and Protocols level
   15 MQCXP-FAPLEVEL         PIC S9(9) BINARY.
**    Capability flags
   15 MQCXP-CAPABILITYFLAGS  PIC S9(9) BINARY.
**    Exit number
   15 MQCXP-EXITNUMBER       PIC S9(9) BINARY.
**    Number of bytes in transmission buffer reserved for exit to use
   15 MQCXP-EXITSPACE        PIC S9(9) BINARY.
**    User Id associated with remote certificate
   15  MQCXP-SSLCERTUSERID   PIC X(12).
```

```
** Length of distinguished name of issuer of remote 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).
```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura MQCXP.

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D* MQCXP Structure
     D*
     D* Structure identifier
     D  CXSID                  1      4
     D* Structure version number
     D  CXVER                  5      8I 0
     D* Type of exit
     D  CXXID                  9     12I 0
     D* Reason for invoking exit
     D  CXREA                 13     16I 0
     D* Response from exit
     D  CXRES                 17     20I 0
     D* Secondary response from exit
     D  CXRE2                 21     24I 0
     D* Feedback code
     D  CXFB                  25     28I 0
     D* Maximum segment length
     D  CXMSL                 29     32I 0
     D* Exit user area
     D  CXUA                  33     48
     D* Exit data
     D  CXDAT                 49     80
     D* Number of times the message has been retried
     D  CXMRC                 81     84I 0
     D* Minimum interval in milliseconds after which the put operation
     D* should be retried
     D  CXMRI                 85     88I 0
     D* Reason code from previous attempt to put the message
     D  CXMRR                 89     92I 0
     D* Length of header information
     D  CXHDL                 93     96I 0
     D* Partner Name
     D  CXPNM                 97    144
     D* Negotiated Formats and Protocols level
     D  CXFAP                145    148I 0
     D* Capability flags
     D  CXCAP                149    152I 0
     D* Exit number
     D  CXEXN                153    156I 0
     D* Number of bytes in transmission buffer reserved for exit to use
     D  CXHDL                157    160I 0
     D* User identifier associated with remote 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
```

Declaración de ensamblador System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura MQCXP.

```
MQCXP                   DSECT
MQCXP_STRUCID           DS   CL4   Structure identifier
MQCXP_VERSION           DS   F     Structure version number
MQCXP_EXITID            DS   F     Type of exit
MQCXP_EXITREASON        DS   F     Reason for invoking exit
MQCXP_EXITRESPONSE      DS   F     Response from exit
MQCXP_EXITRESPONSE2     DS   F     Secondary response from exit
MQCXP_FEEDBACK          DS   F     Feedback code
MQCXP_MAXSEGMENTLENGTH  DS   F     Maximum segment length
MQCXP_EXITUSERAREA      DS   XL16  Exit user area
MQCXP_EXITDATA          DS   CL32  Exit data
MQCXP_MSGRETRYCOUNT     DS   F     Number of times the message has been
*                                  retried
MQCXP_MSGRETRYINTERVAL  DS   F     Minimum interval in milliseconds
*                                  after which the put operation should
*                                  be retried
MQCXP_MSGRETRYREASON    DS   F     Reason code from previous attempt to
*                                  put the message
MQCXP_HEADERLENGTH      DS   F     Length of header information
MQCXP_PARTNERNAME       DS   CL48  Partner Name
MQCXP_FAPLEVEL          DS   F     Negotiated Formats and Protocols
*                                  level
MQCXP_CAPABILITYFLAGS   DS   F     Capability flags
MQCXP_EXITNUMBER        DS   F     Exit number
MQCXP_EXITSPACE         DS   F     Number of bytes in transmission
*                                  buffer reserved for exit to use
MQCXP_SSLCERTUSERID     DS   CL12  User identifier associated with
*                                  remote 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-Descriptor de espera de salida

La estructura MQXWD es un parámetro de entrada/salida en la llamada MQXWAIT.

Esta estructura sólo está soportada en z/OS.

Referencia relacionada

“Campos” en la página 1575
Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

“Declaración C” en la página 1575

Esta declaración es la declaración C para la estructura MQXWD.

"Declaración de ensamblador System/390" en la página 1576
Esta declaración es la declaración de ensamblador System/390 para la estructura MQXWD.

Campos

Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

StrucId (MQCHAR4)
Este campo especifica el identificador de estructura.

El valor debe ser:

MQXWD_ID_STRUCD
Identificador de la estructura del descriptor de espera de salida.

Para el lenguaje de programación C, también se define la constante MQXWD_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQXWD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQXWD_STRUC_ID.

Versión (MQLONG)
Este campo especifica el número de versión de la estructura.

El valor debe ser:

MQXWD_VERSION_1
Número de versión para la estructura del descriptor de espera de salida.

El valor inicial de este campo es MQXWD_VERSION_1.

Reserved1 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved2 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved3 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

BCE (MQLONG)
Este campo especifica el bloque de control de sucesos en el que se debe esperar.

Este campo es el bloque de control de sucesos (ECB) en el que se debe esperar. Debe establecerse en cero antes de que se emita la llamada MQXWAIT; cuando se complete correctamente, contendrá el código de publicación.

Este campo es un campo de entrada/salida.

Declaración C

Esta declaración es la declaración C para la estructura MQXWD.

```
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
  MQCHAR4  StrucId;    /* Structure identifier */
  MQLONG   Version;    /* Structure version number */
  MQLONG   Reserved1;  /* Reserved */
  MQLONG   Reserved2;  /* Reserved */
  MQLONG   Reserved3;  /* Reserved */
```

```
  MQLONG   ECB;        /* Event control block to wait on */
};
```

Declaración de ensamblador System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura MQXWD.

```
MQXWD            DSECT
MQXWD_STRUCID    DS   CL4  Structure identifier
MQXWD_VERSION    DS   F    Structure version number
MQXWD_RESERVED1  DS   F    Reserved
MQXWD_RESERVED2  DS   F    Reserved
MQXWD_RESERVED3  DS   F    Reserved
MQXWD_ECB        DS   F    Event control block to wait on
*
MQXWD_LENGTH     EQU  *-MQXWD
                 ORG  MQXWD
MQXWD_AREA       DS   CL(MQXWD_LENGTH)
```

Llamada de salida de carga de trabajo del clúster y estructuras de datos

Esta sección proporciona información de referencia para la salida de carga de trabajo de clúster y las estructuras de datos. Esta es información de interfaz de programación de uso general.

Puede escribir salidas de carga de trabajo de clúster en los siguientes lenguajes de programación:

- C
- Ensamblador System/390 (IBM MQ for z/OS)

La llamada se describe en:

- “MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada” en la página 1577

Los tipos de datos de estructura utilizados por la salida se describen en:

- “MQXCLWLN -Navegar por registros de carga de trabajo de clúster” en la página 1578
- “MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster” en la página 1582
- “MQWDR-Estructura de registro de destino de carga de trabajo de clúster” en la página 1591
- “MQWQR -Estructura de registro de cola de carga de trabajo de clúster” en la página 1595
- “MQWCR -Estructura de registros de clúster de carga de trabajo de clúster” en la página 1600
- z/OS Comportamiento asíncrono de los mandatos CLUSTER en z/OS

A lo largo de esta sección, los atributos de gestor de colas y los atributos de cola se muestran completos. Los nombres equivalentes que se utilizan en los mandatos MQSC se muestran a continuación. Para obtener detalles sobre los mandatos MQSC, consulte Mandatos MQSC.

Tabla 824. atributos del gestor de colas

Nombre completo	Nombre utilizado en MQSC
ClusterWorkloadData	CLWLDATA
ClusterWorkloadExit	CLWLEXIT
ClusterWorkloadLength	CLWLLEN

Tabla 825. Atributos de colas

Nombre completo	Nombre utilizado en MQSC
DefBind	DEFBIND
DefPersistence	DEFPSIST
DefPriority	DEFPRTY

Tabla 825. Atributos de colas (continuación)	
Nombre completo	**Nombre utilizado en MQSC**
InhibitPut	PUT
QDesc	DESCR

Tareas relacionadas

Escritura y compilación de salidas de carga de trabajo de clúster

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada

El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

Nota: El gestor de colas no proporciona ningún punto de entrada denominado MQ_CLUSTER_WORKLOAD_EXIT . En su lugar, el nombre de la salida de carga de trabajo de clúster se define mediante el atributo de gestor de colas ClusterWorkloadExit .

La salida MQ_CLUSTER_WORKLOAD_EXIT está soportada en todas las plataformas.

Sintaxis

```
MQ_CLUSTER_WORKLOAD_EXIT (ExitParms)
```

Referencia relacionada

MQXCLWLN -Navegar por registros de carga de trabajo de clúster
La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

MQWDR-Estructura de registro de destino de carga de trabajo de clúster
La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster
La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

parámetros para MQ_CLUSTER_WORKLOAD_EXIT

Descripción de los parámetros en la llamada MQ_CLUSTER_WORKLOAD_EXIT .

***ExitParms* (MQWXP) -entrada/salida**

Bloque de parámetros de salida.

- La salida establece información en MQWXP para indicar cómo gestionar la carga de trabajo.

Referencia relacionada

Notas de uso
La función que realiza la salida de carga de trabajo de clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT da soporte a dos lenguajes, C y High Level Assembler.

Notas de uso

La función que realiza la salida de carga de trabajo de clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CLUSTER_WORKLOAD_EXIT . Sin embargo, se proporciona un typedef para el nombre MQ_CLUSTER_WORKLOAD_EXIT en el lenguaje de programación C. Utilice typedef para declarar la salida escrita por el usuario, para asegurarse de que los parámetros son correctos.

Referencia relacionada

parámetros para MQ_CLUSTER_WORKLOAD_EXIT
Descripción de los parámetros en la llamada MQ_CLUSTER_WORKLOAD_EXIT .

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT da soporte a dos lenguajes, C y High Level Assembler.

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT da soporte a dos lenguajes, C y High Level Assembler.

Invocación en C

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Sustituya *MQ_CLUSTER_WORKLOAD_EXIT* por el nombre de la función de salida de carga de trabajo del clúster.

Declare los parámetros **MQ_CLUSTER_WORKLOAD_EXIT** como se indica a continuación:

```
MQWXP ExitParms; /* Exit parameter block */
```

Invocación en ensamblador de alto nivel

```
CALL EXITNAME,(EXITPARMS)
```

Declare los parámetros como se indica a continuación:

```
EXITPARMS      CMQWXPA       Exit parameter block
```

Referencia relacionada

parámetros para MQ_CLUSTER_WORKLOAD_EXIT
Descripción de los parámetros en la llamada MQ_CLUSTER_WORKLOAD_EXIT .

Notas de uso
La función que realiza la salida de carga de trabajo de clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

MQXCLWLN -Navegar por registros de carga de trabajo de clúster

La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

La memoria caché de clúster es un área de almacenamiento principal utilizada para almacenar información relacionada con el clúster.

Si la memoria caché de clúster es estática, tiene un tamaño fijo. Si lo establece en dinámico, la memoria caché de clúster puede expandirse según sea necesario.

Establezca el tipo de memoria caché de clúster en STATIC o DYNAMIC utilizando un parámetro del sistema o una macro.

- Multi Utilice el parámetro del sistema ClusterCacheType en Multiplatforms.
- z/OS Utilice el parámetro CLCACHE en la macro CSQ6SYSP en z/OS.

Sintaxis

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada
El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

MQWDR-Estructura de registro de destino de carga de trabajo de clúster
La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster
La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

Parámetros para MQXCLWLN -Navegar por registros de carga de trabajo de clúster

Descripción de los parámetros en la llamada MQXCLWLN .

ExitParms (MQWXP) -entrada/salida

Bloque de parámetros de salida.

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar cómo gestionar la carga de trabajo.

CurrentRecord (MQPTR) -entrada

Dirección del registro actual.

Esta estructura contiene información relacionada con la dirección del registro que está examinando actualmente la salida. El registro debe ser uno de los tipos siguientes:

- Registro de destino de carga de trabajo de clúster (MQWDR)
- Registro de cola de carga de trabajo de clúster (MQWQR)
- Registro de clúster de carga de trabajo de clúster (MQWCR)

NextOffset (MQLONG) -entrada

Desplazamiento del siguiente registro.

Esta estructura contiene información relacionada con el desplazamiento del siguiente registro o estructura. *NextOffset* es el valor del campo de desplazamiento adecuado en el registro actual y debe ser uno de los campos siguientes:

- CampoChannelDefDesplazamiento en MQWDR
- CampoClusterRecDesplazamiento en MQWDR
- CampoClusterRecDesplazamiento en MQWQR
- CampoClusterRecDesplazamiento en MQWCR

NextRecord (MQPTR) -salida
Dirección del siguiente registro o estructura.

Esta estructura contiene información relacionada con la dirección del siguiente registro o estructura. Si *CurrentRecord* es la dirección de un MQWDR, y *NextOffset* es el valor del campo ChannelDefOffset , *NextRecord* es la dirección de la estructura de definición de canal (MQCD).

Si no hay ningún registro o estructura siguiente, el gestor de colas establece *NextRecord* en el puntero nulo y la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_NO_RECORD_AVAILABLE.

CompCode (MQLONG) -salida
Código de terminación.

El código de terminación tiene uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG) -salida
Código de razón que califica CompCode

Si CompCode es MQCC_OK:

MQRC_NONE
(0, X'0000')
No hay ninguna razón para informar.

Si *CompCode* es MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE
(2359, X'0937')
No hay ningún registro disponible. Se ha emitido una llamada MQXCLWLN desde una salida de carga de trabajo de clúster para obtener la dirección del siguiente registro de la cadena. El registro actual es el último registro de la cadena. Acción correctiva: Ninguna.

Si *CompCode* es MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR
(2357, X'0935')
El parámetro **CurrentRecord** no es válido. Se ha emitido una llamada MQXCLWLN desde una salida de carga de trabajo de clúster para obtener la dirección del siguiente registro de la cadena. La dirección especificada por el parámetro **CurrentRecord** no es la dirección de un registro válido.

CurrentRecord debe ser la dirección de un registro de destino, MQWDR, un registro de cola (MQWQR) o un registro de clúster (MQWCR) residiendo en la memoria caché del clúster. Acción correctiva: Asegúrese de que la salida de carga de trabajo del clúster pasa la dirección de un registro válido que reside en la memoria caché del clúster.

MQRC_ENVIRONMENT_ERROR
(2012, X'07DC')
La llamada no es válida en el entorno. Se ha emitido una llamada MQXCLWLN , pero no desde una salida de carga de trabajo de clúster.

MQRC_NEXT_OFFSET_ERROR
(2358, X'0936')
El parámetro **NextOffset** no es válido. Se ha emitido una llamada MQXCLWLN desde una salida de carga de trabajo de clúster para obtener la dirección del siguiente registro de la cadena. El desplazamiento especificado por el parámetro **NextOffset** no es válido. **NextOffset** debe ser el valor de uno de los campos siguientes:

- CampoChannelDefDesplazamiento en MQWDR
- CampoClusterRecDesplazamiento en MQWDR
- CampoClusterRecDesplazamiento en MQWQR
- CampoClusterRecDesplazamiento en MQWCR

Acción correctiva: Asegúrese de que el valor especificado para el parámetro **NextOffset** es el valor de uno de los campos listados anteriormente.

MQRC_NEXT_RECORD_ERROR
(2361, X'0939')
El parámetro **NextRecord** no es válido.

MQRC_WXP_ERROR
(2356, X'0934')
Estructura de parámetro de salida de carga de trabajo no válida. Se ha emitido una llamada MQXCLWLN desde una salida de carga de trabajo de clúster para obtener la dirección del siguiente registro de la cadena. La estructura del parámetro de salida de carga de trabajo **ExitParms** no es válida, por una de las razones siguientes:

- El puntero del parámetro no es válido. No siempre es posible detectar punteros de parámetro que no son válidos; si no se detectan, se producen resultados imprevisibles.
- El campo StrucId no es MQWXP_STRUC_ID.
- El campo Versión no es MQWXP_VERSION_2.
- El campo Contexto no contiene el valor pasado a la salida por el gestor de colas.

Acción correctiva: Asegúrese de que el parámetro especificado para **ExitParms** es la estructura MQWXP que se pasó a la salida cuando se invocó la salida.

Referencia relacionada

Notas de uso para MQXCLWLN-Navegar por registros de carga de trabajo de clúster
Utilice MQXCLWLN para navegar por los registros de clúster, incluso si la memoria caché es estática.

Invocaciones de lenguaje de MQXCLWLN
MQXCLWLN da soporte a dos lenguajes, C y High Level Assembler.

Notas de uso para MQXCLWLN-Navegar por registros de carga de trabajo de clúster

Utilice MQXCLWLN para navegar por los registros de clúster, incluso si la memoria caché es estática.

Si la memoria caché de clúster es dinámica, se debe utilizar la llamada MQXCLWLN para navegar por los registros. La salida finaliza de forma anómala si se utiliza una aritmética simple de puntero y desplazamiento para navegar por los registros.

Si la memoria caché de clúster es estática, no es necesario utilizar MQXCLWLN para navegar por los registros. Normalmente, utilice MQXCLWLN incluso cuando la memoria caché sea estática. A continuación, puede cambiar la memoria caché de clúster para que sea dinámica sin necesidad de cambiar la salida de carga de trabajo.

Referencia relacionada

Parámetros para MQXCLWLN -Navegar por registros de carga de trabajo de clúster
Descripción de los parámetros en la llamada MQXCLWLN .

Invocaciones de lenguaje de MQXCLWLN
MQXCLWLN da soporte a dos lenguajes, C y High Level Assembler.

Invocaciones de lenguaje de MQXCLWLN

MQXCLWLN da soporte a dos lenguajes, C y High Level Assembler.

Invocación en C

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQXCLWLN,(CLWLEXITPARMS,CURRENTRECORD,NEXTOFFSET,NEXTRECORD,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Referencia relacionada

Parámetros para MQXCLWLN -Navegar por registros de carga de trabajo de clúster
Descripción de los parámetros en la llamada MQXCLWLN .

Notas de uso para MQXCLWLN-Navegar por registros de carga de trabajo de clúster
Utilice MQXCLWLN para navegar por los registros de clúster, incluso si la memoria caché es estática.

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

En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

Tabla 826. Campos en MQWXP

Campo	Descripción	Página
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
ExitId	Tipo de salida	ExitId
ExitReason	Razón de la invocación de la salida	ExitReason
ExitResponse	Respuesta de la salida	ExitResponse
ExitResponse2	Respuesta secundaria de salida	ExitResponse2
Feedback	Código de retroalimentación	Observaciones
Flags	Marca los valores. Estos distintivos de bits se utilizan para indicar información sobre el mensaje que se está colocando	Indicadores
ExitUserArea	Salir del área de usuario	ExitUserArea
ExitData	Datos de salida	ExitData
MsgDescPtr	Dirección del descriptor de mensaje (MQMD)	MsgDescPtr
MsgBufferPtr	Dirección del almacenamiento intermedio que contiene algunos o todos los datos del mensaje	MsgBufferPtr

Tabla 826. Campos en MQWXP (continuación)		
Campo	**Descripción**	**Página**
MsgBufferLength	Longitud del almacenamiento intermedio que contiene datos de mensaje	MsgBufferLongitud
MsgLength	Longitud del mensaje completo	MsgLength
QName	Nombre de cola	QName
QMgrName	Nombre del gestor de colas local	QMgrName
DestinationCount	Número de destinos posibles	DestinationCount
DestinationChosen	Destino elegido	DestinationChosen
DestinationArrayPtr	Dirección de una matriz de punteros a registros de destino (MQWDR)	DestinationArrayPtr
QArrayPtr	Dirección de una matriz de punteros a registros de cola (MQWQR)	QArrayPtr
Nota: Los campos restantes se ignoran si la versión es menor que MQWXP_VERSION_2.		
CacheContext	Información de contexto	CacheContext
CacheType	Tipo de memoria caché de clúster	CacheType
Nota: Los campos restantes se ignoran si la versión es menor que MQWXP_VERSION_3.		
CLWLMRUChannels	Número máximo de canales de clúster de salida activos permitidos	CLWLMRUChannels
Nota: Los campos restantes se ignoran si la versión es menor que MQWXP_VERSION_4.		
pEntryPoints	Dirección de la estructura MQIEP para permitir que se realicen llamadas MQI y DCI	PuntospEntry

La estructura de parámetros de salida de carga de trabajo de clúster describe la información que se pasa a la salida de carga de trabajo de clúster.

La estructura de parámetros de salida de carga de trabajo de clúster está soportada en todas las plataformas

Además, las estructuras MQWXP1, MQWXP2 y MQWXP3 están disponibles para compatibilidad con versiones anteriores.

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada
El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

MQXCLWLN -Navegar por registros de carga de trabajo de clúster
La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

MQWDR-Estructura de registro de destino de carga de trabajo de clúster
La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster

La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

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

Descripción de los campos en MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster

StrucId (MQCHAR4)-entrada

El identificador de estructura para la estructura de parámetros de salida de carga de trabajo de clúster.

- El valor de StrucId es MQWXP_STRUC_ID.
- Para el lenguaje de programación C, también se define la constante MQWXP_STRUC_ID_ARRAY . Tiene el mismo valor que MQWXP_STRUC_ID. Es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)-entrada

Indica el número de versión de la estructura. Versión toma uno de los valores siguientes:

MQWXP_VERSION_1

Estructura de parámetros de salida de carga de trabajo de clúster Version-1 .

MQWXP_VERSION_1 está soportado en todos los entornos.

MQWXP_VERSION_2

Estructura de parámetros de salida de carga de trabajo de clúster Version-2 .

MQWXP_VERSION_2 está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

MQWXP_VERSION_3

Estructura de parámetros de salida de carga de trabajo de clúster Version-3 .

MQWXP_VERSION_3 está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

MQWXP_VERSION_4

Estructura de parámetros de salida de carga de trabajo de clúster Version-4 .

MQWXP_VERSION_4 está soportado en los entornos siguientes:

- AIX
- IBM i
- Linux
- Solaris
- Windows

MQWXP_CURRENT_VERSION
Versión actual de la estructura de parámetros de salida de carga de trabajo de clúster.

ExitId (MQLONG)-entrada
Indica el tipo de salida que se llama. La salida de carga de trabajo de clúster es la única salida soportada.

- El valor de `ExitId` debe ser `MQXT_CLUSTER_WORKLOAD_EXIT`

ExitReason (MQLONG)-entrada
Indica la razón por la que se invoca la salida de carga de trabajo del clúster. `ExitReason` toma uno de los valores siguientes:

MQXR_INIT
Indica que la salida se está invocando por primera vez.

Adquiera e inicialice los recursos que pueda necesitar la salida, como el almacenamiento principal.

MQXR_TERM
Indica que la salida está a punto de terminar.

Libere los recursos que la salida pueda haber adquirido desde que se inicializó, como el almacenamiento principal.

MQXR_CLWL_OPEN
Invocado por `MQOPEN`.

MQXR_CLWL_PUT
Invocado por `MQPUT` o `MQPUT1`.

MQXR_CLWL_MOVE
Llamado por MCA cuando el estado del canal ha cambiado.

MQXR_CLWL_REPOS
Invocado por `MQPUT` o `MQPUT1` para un mensaje PCF de gestor de repositorios.

MQXR_CLWL_REPOS_MOVE
Llamado por MCA para un mensaje PCF de gestor de repositorios si el estado del canal ha cambiado.

ExitResponse (MQLONG)-salida
Establezca `ExitResponse` para indicar si el proceso del mensaje continúa. Tiene que ser uno de los valores siguientes:

MQXCC_OK
Continúe procesando el mensaje con normalidad.

- `DestinationChosen` identifica el destino al que se va a enviar el mensaje.

MQXCC_SUPPRESS_FUNCTION
Discontinuar el proceso del mensaje.

- Las acciones realizadas por el gestor de colas dependen de la razón por la que se ha invocado la salida:

Tabla 827. Acciones realizadas por el gestor de colas

ExitReason	Acción realizada
– `MQXR_CLWL_OPEN` – `MQXR_CLWL_REPOS` – `MQXR_CLWL_PUT`	La llamada `MQOPEN`, `MQPUT`o `MQPUT1` falla con el código de terminación `MQCC_FAILED` y el código de razón `MQRC_STOPPED_BY_CLUSTER_EXIT`.
– `MQXR_CLWL_MOVE` – `MQXR_CLWL_REPOS_MOVE`	El mensaje se coloca en la cola de mensajes no entregados.

MQXCC_SUPPRESS_EXIT
Continúe procesando el mensaje actual normalmente. No vuelva a invocar la salida hasta que el gestor de colas concluya.

El gestor de colas procesa los mensajes posteriores como si el atributo de gestor de colas `ClusterWorkloadExit` estuviera en blanco. `DestinationChosen` identifica el destino al que se envía el mensaje actual.

Cualquier otro valor
Procese el mensaje como si se hubiera especificado `MQXCC_SUPPRESS_FUNCTION` .

`ExitResponse2` (`MQLONG`)-entrada/salida
Establezca `ExitResponse2` para proporcionar al gestor de colas más información.

- `MQXR2_STATIC_CACHE` es el valor predeterminado y se establece en la entrada a la salida.
- Cuando `ExitReason` tiene el valor `MQXR_INIT`, la salida puede establecer uno de los valores siguientes en `ExitResponse2`:

  MQXR2_STATIC_CACHE
  La salida requiere una memoria caché de clúster estático.

  – Si la memoria caché de clúster es estática, la salida no necesita utilizar la llamada `MQXCLWLN` para navegar por las cadenas de registros de la memoria caché de clúster.

  – Si la memoria caché de clúster es dinámica, la salida no puede navegar correctamente por los registros de la memoria caché.

  Nota: El gestor de colas procesa la devolución de la llamada `MQXR_INIT` como si la salida hubiera devuelto `MQXCC_SUPPRESS_EXIT` en el campo `ExitResponse` .

  MQXR2_DYNAMIC_CACHE
  La salida puede funcionar con una memoria caché estática o dinámica.

  – Si la salida devuelve este valor, la salida debe utilizar la llamada `MQXCLWLN` para navegar por las cadenas de registros de la memoria caché de clúster.

Feedback (`MQLONG`)-entrada
Un campo reservado. El valor es cero.

`Distintivos` (`MQLONG`)-entrada
Indica información sobre el mensaje que se está colocando.

- El valor de `Distintivos` es `MQWXP_PUT_BY_CLUSTER_CHL`. El mensaje se origina desde un canal de clúster, en lugar de localmente o desde un canal que no es de clúster. En otras palabras, el mensaje procede de otro gestor de colas de clúster.

`Reservado` (`MQLONG`)-entrada
Un campo reservado. El valor es cero.

`ÁreaExitUser` (`MQBYTE16`)-entrada/salida
Establezca `ExitUserArea` para comunicarse entre llamadas a la salida.

- El `ÁreaExitUser` se inicializa en cero binario antes de la primera invocación de la salida. Los cambios realizados en este campo por la salida se conservan en las invocaciones de la salida que se producen entre la llamada `MQCONN` y la llamada `MQDISC` coincidente. El campo se restablece en cero binario cuando se produce la llamada `MQDISC` .
- La primera invocación de la salida se indica mediante el campo `ExitReason` que tiene el valor `MQXR_INIT`.
- Se definen las constantes siguientes:

  MQXUA_NONE -serie
  MQXUA_NONE_ARRAY -matriz de caracteres
  No hay información de usuario. Ambas constantes son binarias cero para la longitud del campo.

  MQ_EXIT_USER_AREA_LENGTH
  La longitud del `Área deExitUser`.

ExitData (MQCHAR32)-entrada
El valor del atributo del gestor de colas ClusterWorkloadData . Si no se ha definido ningún valor para ese atributo, este campo contendrá espacios en blanco.

- La longitud de ExitData la proporciona MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD)-entrada
La dirección de una copia del descriptor de mensaje (MQMD) para el mensaje que se está procesando.

- El gestor de colas ignora los cambios realizados en el descriptor de mensaje por la salida.
- Si ExitReason tiene uno de los valores siguientes MsgDescPtr se establece en el puntero nulo y no se pasa ningún descriptor de mensaje a la salida:
  - MQXR_INIT
  - MQXR_TERM
  - MQXR_CLWL_OPEN

MsgBufferPtr (PMQVOID)-entrada
La dirección de un almacenamiento intermedio que contiene una copia de la primera longitud de MsgBuffer bytes de los datos del mensaje.

- El gestor de colas ignora los cambios realizados en los datos de mensaje por la salida.
- No se pasan datos de mensaje a la salida cuando:
  - MsgDescPtr es el puntero nulo.
  - El mensaje no tiene datos.
  - El atributo del gestor de colas ClusterWorkloadLength es cero.

  En estos casos, MsgBufferPtr es el puntero nulo.

MsgBufferLength (MQLONG)-entrada
La longitud del almacenamiento intermedio que contiene los datos de mensaje pasados a la salida.

- La longitud la controla el atributo de gestor de colas ClusterWorkloadLength .
- La longitud puede ser menor que la longitud del mensaje completo, consulte MsgLength.

MsgLength (MQLONG)-entrada
La longitud del mensaje completo pasado a la salida.

- MsgBufferLongitud puede ser menor que la longitud del mensaje completo.
- MsgLength es cero si ExitReason es MQXR_INIT, MQXR_TERMo MQXR_CLWL_OPEN.

QName (MQCHAR48)-entrada
El nombre de la cola de destino. La cola es una cola de clúster.

- La longitud de QName es MQ_Q_NAME_LENGTH.

QMgrName (MQCHAR48)-entrada
El nombre del gestor de colas local que ha invocado la salida de carga de trabajo de clúster.

- La longitud de QMgrName es MQ_Q_MGR_NAME_LENGTH.

DestinationCount (MQLONG)-entrada
El número de destinos posibles. Los destinos son instancias de la cola de destino y las describen los registros de destino.

- Un registro de destino es una estructura MQWDR . Hay una estructura para cada ruta posible a cada instancia de la cola.
- Las estructuras MQWDR se direccionan mediante una matriz de punteros, consulte DestinationArrayPtr.

DestinationChosen (MQLONG)-entrada/salida
El destino elegido.

- El número de la estructura MQWDR que identifica la ruta y la instancia de cola donde se va a enviar el mensaje.
- El valor está en el rango 1- DestinationCount.
- En la entrada de la salida, DestinationChosen indica la ruta y la instancia de cola que el gestor de colas ha seleccionado. La salida puede aceptar esta opción o elegir una ruta y una instancia de cola diferentes.
- El valor establecido por la salida debe estar en el rango 1- DestinationCount. Si se devuelve cualquier otro valor, el gestor de colas utiliza el valor de DestinationChosen en la entrada de la salida.

DestinationArrayPtr (PPMQWDR)-entrada
La dirección de una matriz de punteros a registros de destino (MQWDR).

- Hay DestinationCount registros de destino.

QArrayPtr (PPMQWQR)-entrada
La dirección de una matriz de punteros a registros de cola (MQWQR).

- Si hay registros de cola disponibles, hay DestinationCount de ellos.
- Si no hay registros de cola disponibles, QArrayPtr es el puntero nulo.

  Nota: QArrayPtr puede ser el puntero nulo incluso cuando DestinationCount es mayor que cero.

CacheContext (MQPTR): Versión 2-entrada
El campo CacheContext está reservado para que lo utilice el gestor de colas. La salida no debe alterar el valor de este campo.

CacheType (MQLONG): Versión 2-entrada
La memoria caché de clúster tiene uno de los tipos siguientes:

MQCLCT_STATIC
La memoria caché es estática.

- El tamaño de la memoria caché es fijo y no puede crecer a medida que opera el gestor de colas.
- No es necesario utilizar la llamada MQXCLWLN para navegar por los registros de este tipo de memoria caché.

MQCLCT_DYNAMIC
La memoria caché es dinámica.

- El tamaño de la memoria caché puede aumentar para dar cabida a la información de clúster variable.
- Debe utilizar la llamada MQXCLWLN para navegar por los registros de este tipo de memoria caché.

CLWLMRUChannels (MQLONG): Versión 3-entrada
Indica el número máximo de canales de clúster de salida activos, que se deben tener en cuenta para que los utilice el algoritmo de elección de carga de trabajo de clúster.

- CLWLMRUChannels es un valor comprendido entre 1 y 999 999 999.

pEntryPoints (PMQIEP): Versión 4
La dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas MQI y DCI.

Referencia relacionada
Valores iniciales y declaraciones de idioma para MQWXP
Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

Valores iniciales y declaraciones de idioma para MQWXP
Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

Tabla 828. Valores iniciales de los campos en MQWXP		
Nombre de campo	**Nombre de constante**	**Valor de constante**
StrucId	MQWXP_STRUC_ID	'WXP¬'
Version	MQWXP_VERSION_2	2
ExitId	Ninguna	0
ExitReason	MQXCC_OK	0
ExitResponse	Ninguna	0
ExitResponse2	Ninguna	0
Flags	Ninguna	0
ExitUserArea	{MQXUA_NONE_ARRAY}	0
ExitData	Ninguna	""
MsgDescPtr	Ninguna	NULL
MsgBufferPtr	Ninguna	NULL
MsgBufferLength	Ninguna	0
MsgBufferPtr	Ninguna	0
QName	Ninguna	""
QMgrName	Ninguna	""
DestinationCount	Ninguna	0
DestinationChosen	Ninguna	0
DestinationArrayPtr	Ninguna	NULL
QArrayPtr	Ninguna	NULL
CacheContext	Ninguna	NULL
CacheType	MQCLCT_DYNAMIC	1
CLWLMRUChannels	Ninguna	0
pEntryPoints	Ninguna	NULL

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macro MQWXP_DEFAULT contiene los valores predeterminados. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

Declaración 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)
```

Referencia relacionada

Campos en MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
Descripción de los campos en MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster

MQWDR-Estructura de registro de destino de carga de trabajo de clúster

La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

Tabla 829. Campos en MQWDR		
Campo	**Descripción**	**Página**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQWDR	StrucLength
QMgrFlags	Distintivos del gestor de colas	QMgrFlags
QMgrIdentifier	Identificador de gestor de colas	QMgrIdentifier
QMgrName	Nombre del gestor de colas	QMgrName
ClusterRecOffset	Desplazamiento lógico del primer registro de clúster (MQWCR)	ClusterRecDesplazamiento
ChannelState	Estado del canal	ChannelState
ChannelDefOffset	Desplazamiento lógico de la estructura de definición de canal (MQCD)	ChannelDefDesplazamiento
Nota: Los campos restantes se ignoran si la versión es menor que MQWDR_VERSION_2.		
DestSeqNumber	Número de secuencia de destino de canal	DestSeq
DestSeqFactor	Factor de secuencia de destino de canal para ponderación	FactorDestSeq

La estructura de registro de destino de carga de trabajo de clúster contiene información relacionada con uno de los destinos posibles para el mensaje. Hay una estructura de registro de destino de carga de trabajo de clúster para cada instancia de la cola de destino.

La estructura de registro de destino de carga de trabajo de clúster está soportada en todos los entornos.

Además, las estructuras MQWDR1 y MQWDR2 están disponibles para la compatibilidad con versiones anteriores.

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada
El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

MQXCLWLN -Navegar por registros de carga de trabajo de clúster
La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster
La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

Campos en MQWDR-Estructura de registro de destino de carga de trabajo de clúster

Descripción de los parámetros en MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

StrucId (MQCHAR4) -entrada
Identificador de estructura para la estructura de registro de destino de carga de trabajo de clúster.

- El valor de StrucId es MQWDR_STRUC_ID.
- Para el lenguaje de programación C, también se define la constante MQWDR_STRUC_ID_ARRAY . Tiene el mismo valor que MQWDR_STRUC_ID. Es una matriz de caracteres en lugar de una serie.

Versión (MQLONG) -entrada
El número de versión de la estructura. Versión toma uno de los valores siguientes:

MQWDR_VERSION_1
Registro de destino de carga de trabajo de clúster Version-1 .

MQWDR_VERSION_2
Registro de destino de carga de trabajo de clúster Version-2 .

MQWDR_CURRENT_VERSION
Versión actual del registro de destino de carga de trabajo de clúster.

StrucLength (MQLONG) -entrada
La longitud de la estructura MQWDR . StrucLength toma uno de los valores siguientes:

MQWDR_LENGTH_1
Longitud del registro de destino de carga de trabajo de clúster version-1 .

MQWDR_LENGTH_2
Longitud del registro de destino de carga de trabajo de clúster version-2 .

MQWDR_CURRENT_LENGTH
Longitud de la versión actual del registro de destino de carga de trabajo de clúster.

QMgrFlags (MQLONG) -entrada
El gestor de colas señala las propiedades del gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR . Se definen los distintivos siguientes:

MQQMF_REPOSITORY_Q_MGR
El destino es un gestor de colas de repositorio completo.

MQQMF_CLUSSDR_USER_DEFINED
El canal de clúster emisor se ha definido manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED
El canal de clúster emisor se ha definido automáticamente.

MQQMF_AVAILABLE
El gestor de colas de destino está disponible para recibir mensajes.

Otros valores
El gestor de colas puede establecer otros distintivos en el campo para fines internos.

QMgrIdentifier (MQCHAR48) -entrada
El identificador del gestor de colas es un identificador exclusivo para el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR .

- El identificador lo genera el gestor de colas.
- La longitud de QMgrIdentifier es MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) -entrada
El nombre del gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR .

- QMgrName puede ser el nombre del gestor de colas local, así como otro gestor de colas del clúster.
- La longitud de QMgrName es MQ_Q_MGR_NAME_LENGTH.

ClusterRecDesplazamiento (MQLONG) -entrada
El desplazamiento lógico de la primera estructura MQWCR que pertenece a la estructura MQWDR .

- Para memorias caché estáticas, `ClusterRecDesplazamiento` es el desplazamiento de la primera estructura `MQWCR` que pertenece a la estructura `MQWDR` .
- El desplazamiento se mide en bytes desde el inicio de la estructura `MQWDR` .
- No utilice el desplazamiento lógico para la aritmética de puntero con memorias caché dinámicas. Para obtener la dirección del siguiente registro, se debe utilizar la llamada `MQXCLWLN` .

ChannelState (MQLONG) -entrada
El estado del canal que enlaza el gestor de colas local con el gestor de colas identificado por la estructura `MQWDR` . Son posibles los siguientes valores:

MQCHS_BINDING
El canal está negociando con el socio.

MQCHS_INACTIVE
El canal no está activo.

MQCHS_INITIALIZING
El canal se está inicializando.

MQCHS_PAUSED
El canal se ha detenido.

MQCHS_REQUESTING
El canal peticionario está solicitando conexión.

MQCHS_RETRYING
El canal está reintentándose para establecer la conexión.

MQCHS_RUNNING
El canal está transfiriendo o esperando mensajes.

MQCHS_STARTING
El canal está a la espera de activarse.

MQCHS_STOPPING
El canal se está deteniendo.

MQCHS_STOPPED
El canal se ha detenido.

ChannelDefDesplazamiento (MQLONG) -entrada
El desplazamiento lógico de la definición de canal (`MQCD`) para el canal que enlaza el gestor de colas local con el gestor de colas identificado por la estructura `MQWDR` .

- `ChannelDefDesplazamiento` es como `ClusterRecDesplazamiento`
- El desplazamiento lógico no se puede utilizar en aritmética de puntero. Para obtener la dirección del siguiente registro, se debe utilizar la llamada `MQXCLWLN` .

FactorDestSeq (MQLONG) -entrada
El factor de secuencia de destino que permite una elección del canal basada en el peso.

- `FactorDestSeq` se utiliza antes de que el gestor de colas lo cambie.
- El gestor de carga de trabajo aumenta `DestSeqFactor` de una forma que garantiza que los mensajes se distribuyan en canales inactivos según su peso.

DestSeqNúmero (MQLONG) -entrada
El valor de destino de canal de clúster antes de que el gestor de colas lo cambie.

- El gestor de carga de trabajo aumenta `DestSeqNúmero` cada vez que se coloca un mensaje en ese canal.
- Las salidas de carga de trabajo pueden utilizar `DestSeqNúmero` para decidir qué canal debe dejar un mensaje inactivo.

Referencia relacionada
Valores iniciales y declaraciones de lenguaje para MQWDR

Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWDR -Registro de destino de carga de trabajo de clúster.

Valores iniciales y declaraciones de lenguaje para MQWDR

Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWDR -Registro de destino de carga de trabajo de clúster.

Tabla 830. Valores iniciales de los campos en MQWDR

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQWDR_STRUC_ID	'WDR¬'
Version	MQWDR_VERSION_1	1
StrucLength	MQWDR_CURRENT_LENGTH [3]	136
QMgrFlags	MQWDR_NONE	0
QMgrIdentifier	Ninguna	""
QMgrName	Ninguna	""
ClusterRecOffset	Ninguna	0
ChannelState	Ninguna	0
ChannelDefOffset	Ninguna	0
DestSeqNumber	Ninguna	0
DestSeqFactor	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macro MQWDR_DEFAULT contiene los valores predeterminados. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQWDR MyWDR = {MQWDR_DEFAULT};
```

3. Los valores iniciales establecen intencionadamente la longitud de la estructura en la longitud de la versión actual y no en la versión 1 de la estructura.

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)
```

Declaración 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 */
};
```

Referencia relacionada

Campos en MQWDR-Estructura de registro de destino de carga de trabajo de clúster
Descripción de los parámetros en MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster

En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

Tabla 831. Campos en MQWQR		
Campo	**Descripción**	**Página**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQWQR	StrucLength
QFlags	Distintivos de cola	QFlags
QName	Nombre de cola	QName
QMgrIdentifier	Identificador de gestor de colas	QMgrIdentifier
ClusterRecOffset	Desplazamiento del primer registro de clúster (MQWCR)	ClusterRecDesplazamiento
QType	Tipo de cola	QType
QDesc	Descripción de la cola	QDesc
DefBind	Enlace predeterminado	DefBind
DefPersistence	Persistencia de mensajes predeterminada	DefPersistence
DefPriority	Prioridad de mensajes predeterminada	DefPriority
InhibitPut	Si se permiten operaciones de colocación en la cola	InhibitPut
Nota: Los campos restantes se ignoran si la versión es menor que MQWQR_VERSION_2.		
CWLQueuePriority	Un valor de 0 a 9 que representa la prioridad de la cola	CLWLQueuePriority
CLWLQueueRank	Un valor de 0 a 9 que representa el rango de la cola	CLWLQueueRank
Nota: Los campos restantes se ignoran si la versión es menor que MQWQR_VERSION_3.		
DefPutResponse	Resp predet de transferencia	Respuesta deDefPut

La estructura de registro de cola de carga de trabajo de clúster contiene información relacionada con uno de los posibles destinos del mensaje. Hay una estructura de registro de cola de carga de trabajo de clúster para cada instancia de la cola de destino.

La estructura de registro de cola de carga de trabajo de clúster está soportada en todos los entornos.

Además, las estructuras MQWQR1 y MQWQR2 están disponibles para la compatibilidad con versiones anteriores.

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada
El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

MQXCLWLN -Navegar por registros de carga de trabajo de clúster
La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

MQWDR-Estructura de registro de destino de carga de trabajo de clúster
La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster
La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

Campos en MQWQR -Estructura de registro de cola de carga de trabajo de clúster

Descripción de los campos en MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

StrucId (MQCHAR4) -entrada
Identificador de estructura para la estructura de registro de cola de carga de trabajo de clúster.

- El valor de StrucId es MQWQR_STRUC_ID.
- Para el lenguaje de programación C, también se define la constante MQWQR_STRUC_ID_ARRAY . Tiene el mismo valor que MQWQR_STRUC_ID. Es una matriz de caracteres en lugar de una serie.

Versión (MQLONG) -entrada
El número de versión de la estructura. Versión toma uno de los valores siguientes:

MQWQR_VERSION_1
Registro de cola de carga de trabajo de clúster Version-1 .

MQWQR_VERSION_2
Registro de cola de carga de trabajo de clúster Version-2 .

MQWQR_VERSION_3
Registro de cola de carga de trabajo de clúster Version-3 .

MQWQR_CURRENT_VERSION
Versión actual del registro de cola de carga de trabajo de clúster.

StrucLength (MQLONG) -entrada
Longitud de la estructura MQWQR . StrucLength toma uno de los valores siguientes:

MQWQR_LENGTH_1
Longitud del registro de cola de carga de trabajo de clúster version-1 .

MQWQR_LENGTH_2
Longitud del registro de cola de carga de trabajo de clúster version-2 .

MQWQR_LENGTH_3
Longitud del registro de cola de carga de trabajo de clúster version-3 .

MQWQR_CURRENT_LENGTH
Longitud de la versión actual del registro de cola de carga de trabajo de clúster.

QFlags (MQLONG) -entrada
Los distintivos de cola indican las propiedades de la cola. Se definen los distintivos siguientes:

MQQF_LOCAL_Q
El destino es una cola local.

MQQF_CLWL_USEQ_ANY
Permitir el uso de colas locales y remotas en colocaciones.

MQQF_CLWL_USEQ_LOCAL
Permitir sólo las colocaciones de cola local.

Otros valores
El gestor de colas puede establecer otros distintivos en el campo para fines internos.

QName (MQCHAR48) -entrada
El nombre de la cola que es uno de los destinos posibles del mensaje.

- La longitud de QName es MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) -entrada
El identificador del gestor de colas es un identificador exclusivo para el gestor de colas que aloja la instancia de la cola descrita por la estructura MQWQR .

- El identificador lo genera el gestor de colas.
- La longitud de QMgrIdentifier es MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecDesplazamiento (MQLONG) -entrada
El desplazamiento lógico de la primera estructura MQWCR que pertenece a la estructura MQWQR .

- Para memorias caché estáticas, ClusterRecDesplazamiento es el desplazamiento de la primera estructura MQWCR que pertenece a la estructura MQWQR .
- El desplazamiento se mide en bytes desde el inicio de la estructura MQWQR .
- No utilice el desplazamiento lógico para la aritmética de puntero con memorias caché dinámicas. Para obtener la dirección del siguiente registro, se debe utilizar la llamada MQXCLWLN .

QType (MQLONG) -entrada
El tipo de cola de la cola de destino. Son posibles los siguientes valores:

MQCQT_LOCAL_Q
Cola local.

MQCQT_ALIAS_Q
Cola alias.

MQCQT_REMOTE_ Q
Cola remota.

MQCQT_Q_MGR_ALIAS
Alias de gestor de colas.

QDesc (MQCHAR64) -entrada
El atributo de cola de descripción de cola definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR .

- La longitud de QDesc es MQ_Q_DESC_LENGTH.

DefBind (MQLONG) -entrada
El atributo de cola de enlace predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR . Se debe especificar MQBND_BIND_ON_OPEN o MQBND_BIND_ON_GROUP cuando se utilizan grupos con clústeres. Son posibles los valores siguientes:

MQBND_BIND_ON_OPEN
Enlace arreglado por la llamada MQOPEN .

MQBND_BIND_NOT_FIXED
Enlace no arreglado.

MQBND_BIND_ON_GROUP
Permite a una aplicación solicitar que un grupo de mensajes se asigne a la misma instancia de destino.

DefPersistence (MQLONG) -entrada
El atributo de cola de persistencia de mensajes predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR . Son posibles los siguientes valores:

MQPER_PERSISTENT
El mensaje es persistente.

MQPER_NOT_PERSISTENT
El mensaje no es persistente.

DefPriority (MQLONG) -entrada

El atributo de cola de prioridad de mensajes predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR . El rango de prioridad es 0-MaxPriority.

- 0 es la prioridad más baja.
- MaxPriority es el atributo de gestor de colas del gestor de colas que aloja esta instancia de la cola de destino.

InhibitPut (MQLONG) -entrada
El atributo de cola de colocación inhibida definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR . Son posibles los siguientes valores:

MQQA_PUT_INHIBITED
Las operaciones de colocación están inhibidas.

MQQA_PUT_ALLOWED
Las operaciones de colocación están permitidas.

CLWLQueuePriority (MQLONG) -entrada
El atributo de prioridad de cola de carga de trabajo de clúster definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR .

CLWLQueueRank (MQLONG) -entrada
El rango de cola de carga de trabajo de clúster definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR .

RespuestaDefPut (MQLONG) -entrada
El atributo de cola de respuestas de colocación predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR . Son posibles los siguientes valores:

MQPRT_SYNC_RESPONSE
Respuesta síncrona a llamadas MQPUT o MQPUT1 .

MQPRT_ASYNC_RESPONSE
Respuesta asíncrona a llamadas MQPUT o MQPUT1 .

Referencia relacionada
Valores iniciales y declaraciones de lenguaje para MQWQR
Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWQR -Registro de cola de carga de trabajo de clúster.

Valores iniciales y declaraciones de lenguaje para MQWQR
Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWQR -Registro de cola de carga de trabajo de clúster.

Tabla 832. Valores iniciales de los campos en MQWQR		
Nombre de campo	**Nombre de constante**	**Valor de constante**
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR¬'
Version	MQWQR_VERSION_1	1
StrucLength	MQWQR_CURRENT_LENGTH [3]	212
QFlags	Ninguna	0
QName	Ninguna	""
QMgrIdentifier	Ninguna	""
ClusterRecOffset	Ninguna	0
QType	Ninguna	0
QDesc	Ninguna	""
DefBind	Ninguna	0
DefPersistence	Ninguna	0
DefPriority	Ninguna	0
InhibitPut	Ninguna	0
CLWLQueuePriority	Ninguna	0
CLWLQueueRank	Ninguna	0
DefPutResponse	Ninguna	1

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macro MQWQR_DEFAULT contiene los valores predeterminados. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. Los valores iniciales establecen intencionadamente la longitud de la estructura en la longitud de la versión actual y no en la versión 1 de la estructura.

Declaración 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)
```

Referencia relacionada

Campos en MQWQR -Estructura de registro de cola de carga de trabajo de clúster
Descripción de los campos en MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

MQWCR -Estructura de registros de clúster de carga de trabajo de clúster

La tabla siguiente resume los campos de la estructura de registro de carga de trabajo de clúster MQWCR .

Tabla 833. Campos en MQWCR

Campo	Descripción	Página
ClusterName	Nombre del clúster	ClusterName
ClusterRecOffset	Desplazamiento del siguiente registro de clúster (MQWCR)	ClusterRecDesplazamiento
ClusterFlags	Distintivos de clúster	ClusterFlags

La estructura de registro de clúster de carga de trabajo de clúster contiene información sobre un clúster. Para cada clúster al que pertenece la cola de destino, hay una estructura de registro de clúster de carga de trabajo de clúster.

La estructura de registro de clúster de carga de trabajo de clúster está soportada en todos los entornos.

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT -Descripción de llamada
El gestor de colas llama a la salida de carga de trabajo de clúster para direccionar un mensaje a un gestor de colas disponible.

MQXCLWLN -Navegar por registros de carga de trabajo de clúster
La llamada MQXCLWLN se utiliza para navegar por las cadenas de registros MQWDR, MQWQRy MQWCR almacenados en la memoria caché del clúster.

MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWXP -Estructura de parámetros de salida de carga de trabajo de clúster.

MQWDR-Estructura de registro de destino de carga de trabajo de clúster

La tabla siguiente resume los campos de MQWDR -Estructura de registro de destino de carga de trabajo de clúster.

MQWQR -Estructura de registro de cola de carga de trabajo de clúster
En la tabla siguiente se resumen los campos de MQWQR -Estructura de registro de cola de carga de trabajo de clúster.

Campos en MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.

Descripción de los campos en MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.

ClusterName (MQCHAR48) -entrada
El nombre de un clúster al que pertenece la instancia de la cola de destino que es propietaria de la estructura MQWCR . La instancia de cola de destino se describe mediante una estructura MQWDR .

- La longitud de ClusterName es MQ_CLUSTER_NAME_LENGTH.

ClusterRecDesplazamiento (MQLONG) -entrada
El desplazamiento lógico de la siguiente estructura MQWCR .

- Si no hay más estructuras MQWCR , ClusterRecOffset es cero.
- El desplazamiento se mide en bytes desde el inicio de la estructura MQWCR .

ClusterFlags (MQLONG) -entrada
Los distintivos de clúster indican las propiedades del gestor de colas identificado por la estructura MQWCR . Se definen los distintivos siguientes:

MQQMF_REPOSITORY_Q_MGR
El destino es un gestor de colas de repositorio completo.

MQQMF_CLUSSDR_USER_DEFINED
El canal de clúster emisor se ha definido manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED
El canal de clúster emisor se ha definido automáticamente.

MQQMF_AVAILABLE
El gestor de colas de destino está disponible para recibir mensajes.

Otros valores
El gestor de colas puede establecer otros distintivos en el campo para fines internos.

Referencia relacionada
Valores iniciales y declaraciones de lenguaje para MQWCR
Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.

Valores iniciales y declaraciones de lenguaje para MQWCR

Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.

Tabla 834. Valores iniciales de los campos en MQWCR

Nombre de campo	Nombre de constante	Valor de constante
ClusterName	Ninguna	" "
ClusterRecOffset	Ninguna	0
ClusterFlags	Ninguna	0

Declaración 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)
```

Referencia relacionada

Campos en MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.
Descripción de los campos en MQWCR -Estructura de registro de clúster de carga de trabajo de clúster.

Referencia a la salida de la API

Esta sección proporciona información de referencia principalmente de interés para un programador que escribe salidas de API.

Notas de uso general

notas:

1. Todas las funciones de salida pueden emitir la llamada MQXEP; esta llamada se ha diseñado específicamente para su uso desde las funciones de salida de API.
2. La función MQ_INIT_EXIT no puede emitir ninguna llamada MQ que no sea MQXEP.
3. No puede emitir la llamada MQDISC para la conexión actual.
4. Si una función de salida emite la llamada MQCONN, o la llamada MQCONNX con la opción MQCNO_HANDLE_SHARE_NONE, la llamada se completa con el código de razón MQRC_ALREADY_CONNECTED, y el descriptor de contexto devuelto es el mismo que el que se ha pasado a la salida como parámetro.
5. En general, cuando una función de salida de API emite una llamada MQI, no se llama a las salidas de API de forma recursiva. Sin embargo, si una función de salida emite la llamada MQCONNX con las opciones MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, la llamada devuelve un nuevo descriptor de contexto compartido. Esto proporciona a la suite de salida un descriptor de conexión propio y, por lo tanto, una unidad de trabajo independiente de la unidad de trabajo de la aplicación. La suite de salida puede utilizar este descriptor de contexto para colocar y obtener mensajes dentro de su propia unidad de trabajo, y confirmar o restituir dicha unidad de trabajo; todo esto se puede realizar sin afectar a la unidad de trabajo de la aplicación de ninguna manera.

   Debido a que la función de salida está utilizando un descriptor de conexión que es diferente del descriptor utilizado por la aplicación, las llamadas de MQ emitidas por la función de salida dan como resultado que se invoquen las funciones de salida de API relevantes. Por lo tanto, las funciones de salida se pueden invocar de forma recursiva. Tenga en cuenta que tanto el campo *ExitUserArea* en MQAXP como el área de cadena de salida tienen ámbito de descriptor de conexión. En consecuencia, una función de salida no puede utilizar esas áreas para indicar a otra instancia de sí misma invocada recursivamente que ya está activa.
6. Las funciones de salida también pueden colocar y obtener mensajes dentro de la unidad de trabajo de la aplicación. Cuando la aplicación confirma o restituye la unidad de trabajo, todos los mensajes de la unidad de trabajo se confirman o se restituyen juntos, independientemente de quién los haya colocado en la unidad de trabajo (función de aplicación o salida). Sin embargo, la salida puede hacer que la aplicación supere los límites del sistema antes de lo que sería el caso (por ejemplo, superando el número máximo de mensajes no confirmados en una unidad de trabajo).

Cuando una función de salida utiliza la unidad de trabajo de la aplicación de esta forma, la función de salida normalmente debe evitar emitir la llamada MQCMIT, ya que esto confirma la unidad de trabajo de la aplicación y puede afectar al funcionamiento correcto de la aplicación. Sin embargo, es posible que a veces la función de salida tenga que emitir la llamada MQBACK, si la función de salida encuentra un error grave que impide que se confirme la unidad de trabajo (por ejemplo, un error al colocar un mensaje como parte de la unidad de trabajo de la aplicación). Cuando se llame a MQBACK, asegúrese de que no se cambien los límites de la unidad de trabajo de la aplicación. En esta situación, la función de salida debe establecer los valores adecuados para asegurarse de que el código de terminación MQCC_WARNING y el código de razón MQRC_BACKED_OUT se devuelven a la aplicación, para que la aplicación pueda detectar el hecho de que la unidad de trabajo se ha restituido.

Si una función de salida utiliza el descriptor de conexión de la aplicación para emitir llamadas MQ , estas llamadas no dan como resultado más invocaciones de las funciones de salida de API.

7. Si una función de salida MQXR_BEFORE termina de forma anómala, es posible que el gestor de colas pueda recuperarse de la anomalía. Si es posible, el gestor de colas continúa el proceso como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación finaliza.
8. Si una función de salida MQXR_AFTER termina de forma anómala, es posible que el gestor de colas pueda recuperarse de la anomalía. Si es posible, el gestor de colas continúa el proceso como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación finaliza. Tenga en cuenta que en el último caso, los mensajes recuperados fuera de una unidad de trabajo se pierden (esta es la misma situación que la aplicación que falla inmediatamente después de eliminar un mensaje de la cola).
9. El proceso MCA realiza una confirmación de dos fases.

   Si una salida de API intercepta un MQCMIT de un proceso MCA preparado e intenta realizar una acción dentro de la unidad de trabajo, la acción fallará con el código de razón MQRC_UOW_NOT_AVAILABLE.
10. Para un entorno de varias instalaciones, la única forma de tener una salida que funcione con IBM WebSphere MQ 7.0 y IBM WebSphere MQ 7.1 es escribir la salida de una forma que enlace en IBM WebSphere MQ 7.0 con mqm.Lib y, para salidas no primarias o reubicadas, para asegurarse de que la aplicación encuentra la mqm.Lib correcta para la instalación con la que el gestor de colas está asociado actualmente, antes del inicio de la aplicación. (Por ejemplo, ejecute el mandato **setmqenv -m QM** antes de iniciar la aplicación, incluso si el gestor de colas es propiedad de una instalación de IBM WebSphere MQ 7.0 .)
11. Cuando haya varias instalaciones de IBM MQ disponibles, utilice las salidas escritas para una versión anterior de IBM MQ, ya que es posible que la nueva funcionalidad añadida en la versión posterior no funcione con las versiones anteriores. Para obtener más información sobre los cambios entre releases, consulte Cambios en IBM MQ 8.0.

Estructura de parámetros de salida de API de IBM MQ (MQAXP)

La estructura MQAXP, un bloque de control externo, se utiliza como parámetro de entrada o salida para la salida de API. Este tema también proporciona información sobre cómo los gestores de colas procesan las funciones de salida.

MQAXP tiene la siguiente declaración C:

```
typedef struct tagMQAXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Exit Identifier */
  MQLONG    ExitReason;        /* Exit invocation reason */
  MQLONG    ExitResponse;      /* Response code from exit */
  MQLONG    ExitResponse2;     /* Secondary response code from exit */
  MQLONG    Feedback;          /* Feedback code from exit */
  MQLONG    APICallerType;     /* MQSeries API caller type */
  MQBYTE16  ExitUserArea;      /* User area for use by exit */
  MQCHAR32  ExitData;          /* Exit data area */
  MQCHAR48  ExitInfoName;      /* Exit information name */
```

```
  MQBYTE48  ExitPDArea;        /* Problem determination area */
  MQCHAR48  QMgrName;          /* Name of local queue manager */
  PMQACH    ExitChainAreaPtr;  /* Inter exit communication area */
  MQHCONFIG Hconfig;           /* Configuration handle */
  MQLONG    Function;          /* Function Identifier */
  /* Ver:1 */
  MQHMSG    ExitMsgHandle      /* Exit message handle
  /* Ver:2 */
};
```

La siguiente lista de parámetros se pasa cuando se invocan las funciones de una salida de API:

StrucId (MQCHAR4)-entrada
El identificador de estructura de parámetro de salida, con un valor de:

```
MQAXP_STRUC_ID.
```

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada
El número de versión de la estructura, con un valor de:

MQAXP_VERSION_1
Estructura de parámetros de salida de API de la versión 1.

MQAXP_VERSION_2
Estructura de parámetros de salida de API de la versión 2.

VERSIÓN_ACTUAL_MQAXP
Número de versión actual para la estructura de parámetros de salida de API.

El manejador de salida establece este campo en la entrada para cada función de salida.

ExitId (MQLONG)-entrada
El identificador de salida, establecido en la entrada de la rutina de salida, que indica el tipo de salida:

MQXT_API_EXIT
Salida de API.

ExitReason (MQLONG)-entrada
La razón por la que se invoca la salida, establecida en la entrada de cada función de salida:

MQXR_CONNECTION
La salida se invoca para inicializarse a sí misma antes de una llamada MQCONN o MQCONNX, o para finalizar a sí misma después de una llamada MQDISC.

MQXR_ANTES
La salida se invoca antes de ejecutar una llamada de API, o antes de convertir datos en un MQGET.

MQXR_AFTER
La salida se está invocando después de ejecutar una llamada de API.

ExitResponse (MQLONG)-salida
La respuesta de la salida, inicializada al entrar en cada función de salida para:

MQXCC_Correcto
Continúe con normalidad.

La función de salida debe establecer este campo para comunicar al gestor de colas el resultado de la ejecución de la función de salida. El valor debe ser uno de los siguientes:

MQXCC_Correcto
La función de salida se ha completado satisfactoriamente. Continúe con normalidad.

Este valor lo pueden establecer todas las funciones de salida MQXR_ *. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

MQXCC_FAILED
La función de salida ha fallado debido a un error.

Este valor lo pueden establecer todas las funciones de salida MQXR_ *. El gestor de colas establece CompCode en MQCC_FAILED y Reason en:

- MQRC_API_EXIT_INIT_ERROR si la función es MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR si la función es MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR para todas las demás funciones de salida

Los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena.

ExitResponse2 se ignora; el gestor de colas continúa el proceso como si se hubiera devuelto MQXR2_SUPPRESS_CHAIN .

MQXCC_SUPPRESS_FUNCTION
Suprimir la función de API de IBM MQ .

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omite la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. El gestor de colas establece CompCode en MQCC_FAILED, y Reason en MQRC_SUPPRESSED_BY_EXIT, pero los valores establecidos se pueden modificar mediante una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_SKIP_FUNCTION
Omita la función de API de IBM MQ .

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omite la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_SUPPRESS_EXIT
Suprimir todas las funciones de salida que pertenecen al conjunto de salidas.

Este valor sólo lo pueden establecer las funciones de salida MQXR_BEFORE y MQXR_AFTER. Omite *todas* las invocaciones posteriores de las funciones de salida que pertenecen a este conjunto de salidas para esta conexión lógica. Esta omisión continúa hasta que se produce la solicitud de desconexión lógica, cuando se invoca la función MQ_TERM_EXIT con una ExitReason de MQXR_CONNECTION.

La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se ignora.

Si este valor lo establece una función de salida MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte “Cómo procesan los gestores de colas las funciones de salida” en la página 1608.

ExitResponse2 (MQLONG)-salida
Este es un código de respuesta de salida secundario que califica el código de respuesta de salida primario para las funciones de salida MQXR_BEFORE. Se inicializa para:

```
MQXR2_DEFAULT_CONTINUATION
```

en la entrada a una función de salida de llamada de API de IBM MQ . A continuación, se puede establecer en uno de los valores:

MQXR2_DEFAULT_CONTINUATION
Indica si se debe continuar con la siguiente salida de la cadena, en función del valor de ExitResponse.

Si ExitResponse es MQXCC_SUPPRESS_FUNCTION o MQXCC_SKIP_FUNCTION, omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

De lo contrario, invoque la siguiente salida de la cadena.

MQXR2_SUPPRESS_CHAIN
Suprima la cadena.

Omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER para esta invocación de llamada de API. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN
Continúe con la siguiente salida de la cadena.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte "Cómo procesan los gestores de colas las funciones de salida" en la página 1608.

Comentarios (MQLONG)-entrada/salida
Comunicar códigos de retroalimentación entre invocaciones de función de salida. Se inicializa para:

```
MQFB_NONE (0)
```

antes de invocar la primera función de la primera salida de una cadena.

Las salidas pueden establecer este campo en cualquier valor, incluido cualquier valor MQFB_ * o MQRC_ * válido. Las salidas también pueden establecer este campo en un valor de comentarios definido por el usuario en el rango MQFB_APPL_FIRST a MQFB_APPL_LAST.

APICallerType (MQLONG)-entrada
El tipo de interlocutor de API, que indica si el interlocutor de API de IBM MQ es externo o interno del gestor de colas: MQXACT_EXTERNAL o MQXACT_INTERNAL.

ExitUserArea (MQBYTE16)-entrada/salida
Un área de usuario, disponible para todas las salidas asociadas con un objeto ExitInfodeterminado. Se inicializa en MQXUA_NONE (ceros binarios para la longitud del área ExitUser) antes de invocar la primera función de salida (MQ_INIT_EXIT) para el hconn. A partir de entonces, los cambios realizados en este campo por una función de salida se conservan entre invocaciones de funciones de la misma salida.

Este campo está alineado con un múltiplo de 4 MQLONG.

Las salidas también pueden anclar cualquier almacenamiento que asignen desde esta área.

Para cada hconn, cada salida de una cadena de salidas tiene un área ExitUserdiferente. Las salidas de una cadena no pueden compartir el área ExitUsery el contenido del área ExitUserde una salida no está disponible para otra salida de una cadena.

Para programas C, la constante MQXUA_NONE_ARRAY también se define con el mismo valor que MQXUA_NONE, pero como una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32)-entrada
Datos de salida, establecidos en la entrada de cada función de salida en los 32 caracteres de datos específicos de salida que se proporcionan en la salida. Si no define ningún valor en la salida, este campo estará en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

ExitInfoNombre (MQCHAR48)-entrada
El nombre de información de salida, establecido en la entrada de cada función de salida en el ApiExit_name especificado en las definiciones de salida en las stanzas.

ExitPDArea (MQBYTE48)-entrada/salida
Un área de determinación de problemas, inicializada en MQXPDA_NONE (ceros binarios para la longitud del campo) para cada invocación de una función de salida.

Para programas C, la constante MQXPDA_NONE_ARRAY también se define con el mismo valor que MQXPDA_NONE, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida siempre escribe esta área en el rastreo de IBM MQ al final de una salida, incluso cuando la función es satisfactoria.

La longitud de este campo la proporciona MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48)-entrada
El nombre del gestor de colas al que está conectada la aplicación, que ha invocado una salida como resultado del proceso de una llamada de API de IBM MQ .

Si el nombre de un gestor de colas proporcionado en una llamada MQCONN o MQCONNX está en blanco, este campo sigue establecido en el nombre del gestor de colas al que está conectada la aplicación, tanto si la aplicación es servidor como si es cliente.

El manejador de salida establece este campo en la entrada para cada función de salida.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH)-entrada/salida
Se utiliza para comunicar datos entre invocaciones de distintas salidas de una cadena. Se establece en un puntero NULL antes de invocar la primera función (MQ_INIT_EXIT con ExitReason MQXR_CONNECTION) de la primera salida de una cadena de salidas. El valor devuelto por la salida en una invocación se pasa a la siguiente invocación.

Consulte “El área de cadena de salida y la cabecera de área de cadena de salida (MQACH)” en la página 1611 para obtener más detalles sobre cómo utilizar el área de cadena de salida.

Hconfig (MQHCONFIG)-entrada
El descriptor de contexto de configuración, que representa el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas en la función MQ_INIT_EXIT y se pasa posteriormente a la función de salida de API. Se establece en la entrada para cada función de salida.

Puede utilizar Hconfig como puntero a la estructura MQIEP para realizar llamadas MQI y DCI. Debe comprobar que los 4 primeros bytes de HConfig coinciden con el StrucId de la estructura MQIEP antes de utilizar el parámetro HConfig como puntero a la estructura MQIEP.

Función (MQLONG)-entrada
El identificador de función, cuyos valores válidos son las constantes MQXF_ * descritas en “Constantes externas” en la página 1613.

El manejador de salida establece este campo en el valor correcto, al entrar en cada función de salida, en función de la llamada de API IBM MQ que ha dado como resultado la invocación de la salida.

ExitMsgDescriptor de contexto (MQHMSG)-entrada/salida
Cuando la función es MQXF_GET y ExitReason es MQXR_AFTER, se devuelve un descriptor de mensaje válido en este campo que permite a la salida de API acceder a los campos del descriptor de mensaje y a cualquier otra propiedad que coincida con la serie ExitProperties especificada en la estructura MQXEPO al registrar la salida de API.

Cualquier propiedad de descriptor que no sea de mensaje que se devuelva en el descriptor de contexto ExitMsgno estará disponible en el MsgHandle en la estructura MQGMO si se ha especificado una, o en los datos del mensaje.

Cuando la función es MQXF_GET y ExitReason es MQXR_BEFORE, si el programa de salida establece este campo en MQHM_NONE, suprimirá el llenado de las propiedades de manejador ExitMsg.

Este campo no se establece si la versión es menor que MQAXP_VERSION_2.

Cómo procesan los gestores de colas las funciones de salida

El proceso realizado por el gestor de colas al devolver una función de salida depende de ExitResponse y de ExitResponse2.

Tabla 835 en la página 1608 resume las combinaciones posibles y sus efectos para una función de salida MQXR_BEFORE, mostrando:

- Quién establece los parámetros CompCode y Reason de la llamada de API
- Si se invocan las funciones de salida restantes de la cadena MQXR_BEFORE y las funciones de salida coincidentes de la cadena MQXR_AFTER
- Si se invoca la llamada de API

Para una función de salida MQXR_AFTER:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora (las funciones de salida restantes de la cadena MQXR_AFTER siempre se invocan)
- MQXCC_SUPPRESS_FUNCTION y MQXCC_SKIP_FUNCTION no son válidos

Para una función de salida MQXR_CONNECTION:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT no son válidos

En todos los casos, en los que una salida o el gestor de colas establece CompCode y Reason, los valores establecidos se pueden cambiar mediante una salida invocada más adelante, o mediante la llamada de API (si la llamada de API se invoca más tarde).

Tabla 835. Proceso de salida MQXR_BEFORE

Valor de ExitResponse	**CompCode y Reason establecidos por**	**Valor de ExitResponse2 (continuación predeterminada) Cadena**	**Valor de la API ExitResponse2 (continuación predeterminada)**
MQXCC_Correcto	salida	Y	Y
MQXCC_SUPPRESS_EXIT	salida	Y	Y
MQXCC_SUPPRESS_FUNCTION	gestor de colas	N	N
MQXCC_SKIP FUNCIÓN	salida	N	N
MQXCC_FAILED	gestor de colas	N	N

Cómo procesan los clientes las funciones de salida

En general, los clientes procesan las funciones de salida de la misma forma que las aplicaciones de servidor, y el atributo *QMgrName* de esta estructura se aplica tanto si la función está en un servidor como si está en un cliente.

Sin embargo, el cliente no tiene ningún concepto del archivo *mqs.ini* , por lo que las stanzas *ApiExitCommon* y *APIExitTemplate* no se aplican. Solo se aplica la stanza *ApiExitLocal* y esta stanza se configura en el archivo *mqclient.ini* .

Estructura de contexto de salida de API de IBM MQ (MQAXC)

La estructura MQAXC, un bloque de control externo, se utiliza como parámetro de entrada para una salida de API.

MQAXC tiene la siguiente declaración C:

```
typedef struct tagMQAXC {
   MQCHAR4   StrucId;                /* Structure identifier */
   MQLONG    Version;                /* Structure version number */
   MQLONG    Environment;            /* Environment */
   MQCHAR12  UserId;                 /* UserId associated with appl */
   MQBYTE40  SecurityId              /* Extension to UserId running appl */
   MQCHAR264 ConnectionName;         /* Connection name */
   MQLONG    LongMCAUserIdLength;    /* long MCA user identifier length */
   MQLONG    LongRemoteUserIdLength; /* long remote user identifier length */
   MQPTR     LongMCAUserIdPtr;       /* long MCA user identifier address */
   MQPTR     LongRemoteUserIdPtr;    /* long remote user identifier address */
   MQCHAR28  ApplName;               /* Application name */
   MQLONG    ApplType;               /* Application type */
   MQPID     ProcessId;              /* Process identifier */
   MQTID     ThreadId;               /* Thread identifier */

   /* Ver:1 */
   MQCHAR    ChannelName[20]         /* Channel Name */
   MQBYTE4   Reserved1;              /* Reserved */
   PMQCD     pChannelDefinition;     /* Channel Definition pointer */
 };
```

Los parámetros para MQAXC son:

StrucId (MQCHAR4)-entrada
El identificador de estructura de contexto de salida, con un valor de `MQAXC_STRUC_ID`. Para los programas C, también se define la constante MQAXC_STRUC_ID_ARRAY, con el mismo valor que MQAXC_STRUC_ID, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada
El número de versión de la estructura, con un valor de:

MQAXC_VERSION_2
Número de versión para la estructura de contexto de salida.

MQAXC_VERSIÓN_ACTUAL
Número de versión actual para la estructura de contexto de salida.

El manejador de salida establece este campo en la entrada para cada función de salida.

Entorno (MQLONG)-entrada
Entorno desde el que se ha emitido una llamada de API de IBM MQ que ha dado como resultado que se haya controlado una función de salida. Los valores válidos para este campo son:

MQXE_OTHER
Este valor es coherente con las invocaciones que una salida de API ve si se llama a la salida desde una aplicación de servidor. Esto significa que una salida de API se ejecuta sin cambios en un cliente y no ve nada diferente.

Si la salida realmente necesita determinar si se está ejecutando en el cliente, puede hacerlo consultando los campos *ChannelName* y *ChannelDefinition* .

MQXE_MCA
Agente de canal de mensajes

MQXE_MCA_SVRCONN
Un agente de canal de mensajes que actúa en nombre de un cliente

MQXE_COMMAND_SERVER
El servidor de mandatos

MQXE_MQSC
El intérprete de mandatos runmqsc

El manejador de salida establece este campo en la entrada para cada función de salida.

UserId (MQCHAR12)-entrada
El ID de usuario asociado a la aplicación. En particular, en el caso de las conexiones de cliente, este campo contiene el ID de usuario del usuario adoptado en contraposición al ID de usuario bajo el que se ejecuta el código de canal. Si un ID de usuario en blanco fluye desde el cliente, no se realiza ningún cambio en el ID de usuario que ya se está utilizando. Es decir, no se adopta ningún ID de usuario nuevo.

El manejador de salida establece este campo en la entrada para cada función de salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

SecurityId (MQBYTE40)-entrada
Una extensión del ID de usuario que ejecuta la aplicación. Su longitud la proporciona MQ_SECURITY_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

ConnectionName (MQCHAR264)-entrada
El campo de nombre de conexión, establecido en la dirección del cliente. Por ejemplo, para TCP/IP, sería la dirección IP del cliente.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH.

En el caso de un cliente, es la dirección asociada del gestor de colas.

LongMCAUserIdLength (MQLONG)-entrada
Longitud del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario de MCA largo (o cero si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdLength (MQLONG)-entrada
La longitud del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario remoto largo. De lo contrario, este campo se establecerá en cero

En el caso de un cliente, establezca este campo en cero.

LongMCAUserIdPtr (MQPTR)-entrada
Dirección del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario de MCA largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdPtr (MQPTR)-entrada
La dirección del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario remoto largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, establezca este campo en cero.

ApplName (MQCHAR28)-entrada
El nombre de la aplicación o componente que ha emitido la llamada de API IBM MQ .

Las reglas para generar el ApplName son las mismas que para generar el nombre predeterminado para un MQPUT.

El valor de este campo se encuentra consultando el nombre del programa en el sistema operativo. Su longitud la proporciona MQ_APPL_NAME_LENGTH.

ApplType (MQLONG)-entrada
El tipo de aplicación o componente que ha emitido la llamada de API IBM MQ .

El valor es MQAT_DEFAULT para la plataforma en la que se compila la aplicación, o equivale a uno de los valores MQAT_ * definidos.

El manejador de salida establece este campo en la entrada para cada función de salida.

ProcessId (MQPID)-entrada
Identificador de proceso del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ThreadId (MQTID)-entrada
El identificador de hebra de MQ . Es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados FFST , pero puede ser diferente del identificador de hebra del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ChannelName (MQCHAR)-entrada
El nombre del canal, rellenado con espacios en blanco, si es aplicable y conocido.

Si no es aplicable, este campo se establece en caracteres NULL.

Reserved1 (MQBYTE4)-entrada
Este campo está reservado.

ChanneDefinition (PMQCD)-entrada
Puntero a la definición de canal que se utiliza, si es aplicable y conocida.

Si no es aplicable, este campo se establece en caracteres NULL.

Tenga en cuenta que el puntero sólo se completa si la conexión se está procesando en nombre de un canal IBM MQ y esa definición de canal se ha leído.

En concreto, la definición de canal no se proporciona en el servidor cuando se realiza la primera llamada MQCONN para el canal. Además, si el puntero se llena, la estructura (y cualquier subestructura) apuntada por el puntero debe ser tratada como de sólo lectura; cualquier actualización de la estructura conduciría a resultados impredecibles y no está soportada.

En el caso de un cliente, los campos distintos de aquellos con un valor especificado para un cliente, contienen valores que son adecuados para una aplicación cliente.

El área de cadena de salida y la cabecera de área de cadena de salida (MQACH)

Si es necesario, una función de salida puede adquirir almacenamiento para un área de cadena de salida y establecer la ExitChainAreaPtr en MQAXP para que apunte a este almacenamiento.

Las salidas (las mismas o diferentes funciones de salida) pueden adquirir varias áreas de cadena de salida y enlazarlas. Las áreas de cadena de salida sólo deben añadirse o eliminarse de esta lista mientras se llama desde el manejador de salida. Esto garantiza que no haya problemas de serialización causados por distintas hebras que añadan o eliminen áreas de la lista al mismo tiempo.

Un área de cadena de salida debe empezar con una estructura de cabecera MQACH, cuya declaración C es:

```
typedef struct tagMQACH {
   MQCHAR4   StrucId;             /* Structure identifier */
   MQLONG    Version;             /* Structure version number */
   MQLONG    StrucLength;         /* Length of the MQACH structure */
   MQLONG    ChainAreaLength;     /* Exit chain area length */
   MQCHAR48  ExitInfoName         /* Exit information name */
   PMQACH    NextChainAreaPtr;    /* Pointer to next exit chain area */
 };
```

Los campos de la cabecera de área de cadena de salida son:

StrucId (MQCHAR4)-entrada
El identificador de estructura de área de cadena de salida, con un valor inicial, definido por MQACH_DEFAULT, de MQACH_STRUC_ID.

Para programas C, la constante MQACH_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQACH_STRUC_ID, pero como una matriz de caracteres en lugar de una serie.

Versión (MQLONG)-entrada
El número de versión de la estructura, tal como se indica a continuación:

MQACH_VERSION_1
El número de versión para la estructura del parámetro de salida.

VERSIÓN_ACTUAL_MQACH_VERSIÓN
El número de versión actual para la estructura de contexto de salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es `MQACH_CURRENT_VERSION`.

Nota: Si introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Las funciones de salida deben comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la función de salida necesita utilizar.

StrucLength (MQLONG)-entrada
Longitud de la estructura MQACH. Las salidas pueden utilizar este campo para determinar el inicio de los datos de salida, estableciéndolos en la longitud de la estructura creada por la salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es MQACH_CURRENT_LENGTH.

ChainAreaLength (MQLONG)-entrada
La longitud del área de cadena de salida, establecida en la longitud global del área de cadena de salida actual, incluida la cabecera MQACH.

El valor inicial de este campo, definido por MQACH_DEFAULT, es cero.

ExitInfoNombre (MQCHAR48)-entrada
El nombre de información de salida.

Cuando una salida crea una estructura MQACH, debe inicializar este campo con su propio nombre ExitInfo, para que posteriormente esta estructura MQACH pueda ser encontrada por otra instancia de esta salida o por una salida cooperante.

El valor inicial de este campo, definido por MQACH_DEFAULT, es una serie de longitud cero ({""}).

NextChainAreaPtr (PMQACH)-entrada
Puntero al siguiente área de cadena de salida con un valor inicial, definido por MQACH_DEFAULT, de puntero nulo (NULL).

Las funciones de salida deben liberar el almacenamiento para las áreas de cadena de salida que adquieran y manipular los punteros de cadena para eliminar sus áreas de cadena de salida de la lista.

Un área de cadena de salida se puede construir de la siguiente manera:

```
MQAXP.ExitChainAreaPtr ──┐
  ┌──────────────────────┘
  └──→ ┌─────┬───┬────┬──────┬──────────────────────────────────────┐
       │ ACH │ 1 │ 80 │ ──┐  │ First exit's chain area                │
```

```
      └────┴───┴────┴──┴┬┴──────────────────────────────────────┘

┌────────────────────┘

└──→ ┌────┬───┬────┬──┬──────────────────────────┐
     │ ACH │ 1 │ 64 │ ──┐ │ Second exit's chain area  │
     └────┴───┴────┴──┴┬┴──────────────────────────┘

┌────────────────────┘

└──→ ... etc.
```

Constantes externas

Utilice este tema como información de referencia para las constantes externas disponibles para la API.

Las siguientes constantes externas están disponibles para las salidas de API:

MQXF_ * (identificadores de función de salida)

```
MQXF_INIT                      1       X'00000001'
MQXF_TERM                      2       X'00000002'
MQXF_CONN                      3       X'00000003'
MQXF_CONNX                     4       X'00000004'
MQXF_DISC                      5       X'00000005'
MQXF_OPEN                      6       X'00000006'
MQXF_CLOSE                     7       X'00000007'
MQXF_PUT1                      8       X'00000008'
MQXF_PUT                       9       X'00000009'
MQXF_GET                       10      X'0000000A'
MQXF_DATA_CONV_ON_GET          11      X'0000000B'
MQXF_INQ                       12      X'0000000C'
MQXF_SET                       13      X'0000000D'
MQXF_BEGIN                     14      X'0000000E'
MQXF_CMIT                      15      X'0000000F'
MQXF_BACK                      16      X'00000010'
MQXF_STAT                      18      X'00000012'
MQXF_CB                        19      X'00000013'
MQXF_CTL                       20      X'00000014'
MQXF_CALLBACK                  21      X'00000015'
MQXF_SUB                       22      X'00000016'
MQXF_SUBRQ                     23      X'00000017'
MQXF_XACLOSE                   24      X'00000018'
MQXF_XACOMMIT                  25      X'00000019'
MQXF_XACOMPLETE                26      X'0000001A'
MQXF_XAEND                     27      X'0000001B'
MQXF_XAFORGET                  28      X'0000001C'
MQXF_XAOPEN                    29      X'0000001D'
MQXF_XAPREPARE                 30      X'0000001E'
MQXF_XARECOVER                 31      X'0000001F'
MQXF_XAROLLBACK                32      X'00000020'
MQXF_XASTART                   33      X'00000021'
MQXF_AXREG                     34      X'00000022'
MQXF_AXUNREG                   35      X'00000023'
```

MQXR_ * (razones de salida)

```
MQXR_BEFORE                    1       X'00000001'
MQXR_AFTER                     2       X'00000002'
MQXR_CONNECTION                3       X'00000003'
```

MQXE_ * (entornos)

```
MQXE_OTHER                     0       X'00000000'
MQXE_MCA                       1       X'00000001'
MQXE_MCA_SVRCONN               2       X'00000002'
MQXE_COMMAND_SERVER            3       X'00000003'
MQXE_MQSC                      4       X'00000004'
```

MQ* _ * (constantes adicionales)

```
MQAXP_VERSION_1                1
```

```
MQAXP_VERSION_2                 2
MQAXC_VERSION_1                 1
MQACH_VERSION_1                 1

MQAXP_CURRENT_VERSION           1
MQAXC_CURRENT_VERSION           1
MQACH_CURRENT_VERSION           1

MQXACT_EXTERNAL                 1
MQXACT_INTERNAL                 2

MQXT_API_EXIT                   2

MQACH_LENGTH_1                  68 (32-bit platforms)
                                72 (64-bit platforms)
                                80 (128-bit platforms)

MQACH_CURRENT_LENGTH            68 (32-bit platforms)
                                72 (64-bit platforms)
                                80 (128-bit platforms)
```

MQ* _ * (constantes nulas)

```
MQXPDA_NONE                     X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY               '\0','\0',...,'\0','\0'
```

MQXCC_ * (códigos de terminación)

```
MQXCC_FAILED                   -8
```

MQRC_ * (códigos de razón)

MQRC_API_EXIT_ERROR 2374 X'00000946'
Una invocación de función de salida ha devuelto un código de respuesta no válido, o ha fallado de algún modo, y el gestor de colas no puede determinar la siguiente acción a realizar.

Examine los campos ExitResponse y ExitResponse2 de MQAXP para determinar el código de respuesta incorrecto y cambie la salida para que devuelva un código de respuesta válido.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'
El gestor de colas ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'
El gestor de colas ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'
El valor del campo ExitReason proporcionado en una llamada de registro de punto de entrada de salida (MQXEP) es erróneo.

Examine el valor del campo ExitReason para determinar y corregir el valor de razón de salida incorrecta.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'
El valor del campo Reservado es erróneo.

Examine el valor del campo Reservado para determinar y corregir el valor Reservado.

Definiciones de tipo de idioma C

Este tema proporciona información sobre las typedefs asociadas con salidas de API disponibles en lenguaje C.

A continuación se muestran las definiciones de tipo de lenguaje C asociadas con las salidas de API:

```
typedef PMQLONG  MQPOINTER PPMQLONG;
typedef PMQBYTE  MQPOINTER PPMQBYTE;
typedef PMQHOBJ  MQPOINTER PPMQHOBJ;
```

```
typedef PMQOD    MQPOINTER PPMQOD;
typedef PMQMD    MQPOINTER PPMQMD;
typedef PMQPMO   MQPOINTER PPMQPMO;
typedef PMQGMO   MQPOINTER PPMQGMO;
typedef PMQCNO   MQPOINTER PPMQCNO;
typedef PMQBO    MQPOINTER PPMQBO;

typedef MQAXP    MQPOINTER PMQAXP;
typedef MQACH    MQPOINTER PMQACH;
typedef MQAXC    MQPOINTER PMQAXC;

typedef MQCHAR   MQCHAR16[16];
typedef MQCHAR16 MQPOINTER PMQCHAR16;

typedef MQLONG   MQPID;
typedef MQLONG   MQTID;
```

Llamada de registro de punto de entrada de salida (MQXEP)

Utilice esta información para obtener información sobre MQXEP, invocación de lenguaje C MQXEP y prototipo de función C MQXEP.

Utilice la llamada MQXEP para:

1. Registrar los puntos de invocación de salida de API antes y después de IBM MQ en los que invocar las funciones de salida
2. Especificar los puntos de entrada de la función de salida
3. Anular el registro de los puntos de entrada de la función de salida

Normalmente codificaría las llamadas MQXEP en la función de salida MQ_INIT_EXIT, pero puede especificarlas en cualquier función de salida posterior.

Si utiliza una llamada MQXEP para registrar una función de salida ya registrada, la segunda llamada MQXEP se completa correctamente, sustituyendo la función de salida registrada.

Si utiliza una llamada MQXEP para registrar una función de salida NULL, la llamada MQXEP se completa correctamente y se anula el registro de la función de salida.

Si se utilizan llamadas MQXEP para registrar, anular el registro y volver a registrar una función de salida determinada durante la vida de una solicitud de conexión, se reactiva la función de salida registrada anteriormente. Cualquier almacenamiento todavía asignado y asociado con esta instancia de función de salida está disponible para que lo utilicen las funciones de la salida. (Este almacenamiento normalmente se libera durante la invocación de la función de salida de terminación.)

La interfaz con MQXEP es:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

donde:

Hconfig (MQHCONFIG)-entrada
El descriptor de contexto de configuración, que representa la salida de API que incluye el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas inmediatamente antes de invocar la función MQ_INIT_EXIT y se pasa en MQAXP a cada función de salida de API.

ExitReason (MQLONG)-entrada
Razón por la que se está registrando el punto de entrada, por las razones siguientes:

- Inicialización o terminación de nivel de conexión (MQXR_CONNECTION)
- Antes de una llamada de API de IBM MQ (MQXR_BEFORE)
- Después de una llamada de API de IBM MQ (MQXR_AFTER)

Función (MQLONG)-entrada
El identificador de función, cuyos valores válidos son las constantes MQXF_ * (consulte “Constantes externas” en la página 1613).

EntryPoint (PMQFUNC)-entrada
La dirección del punto de entrada para la función de salida que se va a registrar. El valor NULL indica que no se ha proporcionado la función de salida o que se está anulando el registro de un registro anterior de la función de salida.

ExitOpts(MQXEPO)
Las salidas de API pueden especificar opciones que controlan cómo se registran las salidas de API. Si se especifica un puntero nulo para este campo, se asumen los valores predeterminados de la estructura MQXEPO.

CompCode (MQLONG)-salida
El código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
El código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED:

MQRC_HCONFIG_ERROR
(2280, X'8E8') El descriptor de contexto de configuración proporcionado no es válido. Utilice el descriptor de contexto de configuración de MQAXP.

MQRC_EXIT_REASON_ERROR
(2377, X' 949 ') La razón de invocación de función de salida proporcionada no es válida o no es válida para el identificador de función de salida proporcionado.

Utilice una de las razones de invocación de función de salida válidas (valor MQXR_ *) o utilice una combinación válida de identificador de función y razón de salida. (Consulte Tabla 836 en la página 1616.)

MQRC_FUNCTION_ERROR
(2281, X'8E9') El identificador de función proporcionado no es válido por razón de salida de API. La tabla siguiente muestra combinaciones válidas de identificadores de función y ExitReasons.

Tabla 836. Combinaciones válidas de identificadores de función y ExitReasons

Función	ExitReason
MQXF_INIT MQXF_TERM	MQXR_CONNECTION

Tabla 836. Combinaciones válidas de identificadores de función y ExitReasons (continuación)	
Función	**ExitReason**
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CLOSE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_INICIO MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_ANTES MQXR_AFTER
MQXF_DATA_CONV_ON_GET	MQXR_ANTES

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Un intento de registrar o anular el registro de una función de salida ha fallado debido a un problema de recurso.

MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Un intento de registrar o anular el registro de una función de salida ha fallado inesperadamente.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de ExitProperties no válido.

MQRC_XEPO_ERROR
(2507, X'09CB') La estructura de opciones de salida no es válida.

Invocación de lenguaje C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Declaración para lista de parámetros:

```
    MQHCONFIG       Hconfig;        /* Configuration handle */
    MQLONG          ExitReason;     /* Exit reason */
    MQLONG          Function;       /* Function identifier */
    PMQFUNC         EntryPoint;     /* Function entry point */
    MQXEPO          ExitOpts;       /* Options that control the action of MQXEP */
    MQLONG          CompCode;       /* Completion code */
    MQLONG          Reason;         /* Reason code qualifying completion
                                       code */
```

Prototipo de función C MQXEP

```
void MQXEP (
MQHCONFIG    Hconfig,        /* Configuration handle */
MQLONG       ExitReason,     /* Exit reason */
```

```
MQLONG       Function,       /* Function identifier */
PMQFUNC      EntryPoint,     /* Function entry point */
PMQXEPO      pExitOpts;      /* Options that control the action of MQXEP */
PMQLONG      pCompCode,      /* Address of completion code */
PMQLONG      pReason);       /* Address of reason code qualifying completion
                                code */
```

Funciones de salida

Esta sección proporciona información general para ayudarle a utilizar las llamadas de función y describe cómo invocar las funciones de salida individuales.

Utilice esta información para comprender las reglas generales para las rutinas de salida de API y para configurar y limpiar el entorno de ejecución de salida.

Reglas generales para rutinas de salida de API

Se aplican las siguientes reglas generales al invocar rutinas de salida de API:

- En todos los casos, las funciones de salida de API se controlan antes de validar los parámetros de llamada de API y antes de cualquier comprobación de seguridad (en el caso de MQCONN, MQCONNX o MQOPEN).
- Los valores de los campos introducidos y la salida de una rutina de salida son:
  - En la entrada a una *antes de la función de salida de API de* IBM MQ , el valor de un campo puede ser establecido por el programa de aplicación o por una invocación de función de salida anterior.
  - En la salida de una función de salida de API *before* IBM MQ , el valor de un campo se puede dejar sin modificar, o se puede establecer en algún otro valor mediante la función de salida.
  - En la entrada a una función de salida de API *after* IBM MQ , el valor de un campo puede ser el valor establecido por el gestor de colas después de procesar la llamada de API IBM MQ , o puede establecerse en un valor mediante una invocación de función de salida anterior en la cadena de funciones de salida.
  - En la salida de una función de salida de llamada de API *after* IBM MQ , el valor de un campo se puede dejar sin modificar o establecer en algún otro valor mediante la función de salida.
- Las funciones de salida deben comunicarse con el gestor de colas utilizando los campos ExitResponse y ExitResponse2 .
- Los campos CompCode y Código de razón se comunican de nuevo con la aplicación. El gestor de colas y las funciones de salida pueden establecer los campos CompCode y Código de razón.
- La llamada MQXEP devuelve nuevos códigos de razón a las funciones de salida que llaman a MQXEP. Sin embargo, las funciones de salida pueden convertir estos nuevos códigos de razón a cualquier código de razón existente que las aplicaciones existentes y nuevas puedan comprender.
- Cada prototipo de función de salida tiene parámetros similares a la función de la API con un nivel adicional de indirección, excepto CompCode y Reason.
- Las salidas de API pueden emitir llamadas MQI (excepto MQDISC), pero estas llamadas MQI no invocan ellas mismas salidas de API.

Tenga en cuenta que si la aplicación está en un servidor o un cliente, no puede predecir la secuencia de las llamadas de salida de API. Es posible que una llamada BEFORE de salida de API no vaya seguida inmediatamente de una llamada AFTER .

La llamada BEFORE puede ir seguida de otra llamada BEFORE . Por ejemplo:

MQCTL ANTES
ANTES de devolución de llamada
ANTES DE MQPUT
DESPUÉS de MQPUT
DESPUÉS de devolución de llamada
DESPUÉS DE MQCTL

o

ANTES DE XAOPEN
ANTES DE MQCONNX
DESPUÉS de MQCONNX
DESPUÉS DE XAOPEN

En el cliente, hay una salida que puede modificar el comportamiento de la llamada MQCONN o MQCONNX, denominada salida `PreConnect` . La salida `PreConnect` puede modificar cualquiera de los parámetros de la llamada MQCONN o MQCONNX, incluido el nombre del gestor de colas. El cliente llama primero a esta salida y, a continuación, invoca la llamada MQCONN o MQCONNX. Tenga en cuenta que sólo la llamada MQCONN o MQCONNX inicial invoca la salida de API; las llamadas de reconexión posteriores no tienen ningún efecto.

Entorno de ejecución

En general, todos los errores de las funciones de salida se comunican de nuevo al manejador de salida utilizando los campos ExitResponse y ExitResponse2 en MQAXP.

Estos errores a su vez se convierten en valores MQCC_ * y MQRC_ * y se comunican de nuevo a la aplicación en los campos CompCode y Reason. Sin embargo, los errores encontrados en la lógica del manejador de salida se comunican de nuevo a la aplicación como valores MQCC_ * y MQRC_ * en los campos CompCode y Reason.

Si una función MQ_TERM_EXIT devuelve un error:

- La llamada MQDISC ya ha tenido lugar
- No hay otra oportunidad de controlar la función de salida *after* MQ_TERM_EXIT (y, por lo tanto, realizar la limpieza del entorno de ejecución de salida)
- No se ha realizado la limpieza del entorno de ejecución de salida

La salida no se puede descargar porque puede que todavía esté en uso. Además, otras salidas registradas más abajo en la cadena de salida para las que la salida *anterior* ha sido satisfactoria, se activarán en el orden inverso.

Configuración del entorno de ejecución de salida

Al procesar una llamada MQCONN o MQCONNX explícita, la lógica de manejo de salida configura el entorno de ejecución de salida antes de invocar la función de inicialización de salida (MQ_INIT_EXIT). La configuración del entorno de ejecución de salida implica cargar la salida, adquirir almacenamiento para e inicializar las estructuras de parámetros de salida. También se asigna el descriptor de contexto de configuración de salida.

Si se producen errores durante esta fase, la llamada MQCONN o MQCONNX falla con CompCode MQCC_FAILED y uno de los siguientes códigos de razón:

MQRC_API_EXIT_LOAD_ERROR
Ha fallado un intento de cargar un módulo de salida de API.

MQRC_API_EXIT_NOT_FOUND
No se ha podido encontrar una función de salida de API en el módulo de salida de API.

MQRC_STORAGE_NOT_AVAILABLE
Ha fallado un intento de inicializar el entorno de ejecución para una función de salida de API porque no había suficiente almacenamiento disponible.

MQRC_API_EXIT_INIT_ERROR
Se ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

Limpieza del entorno de ejecución de salida

Al procesar una llamada MQDISC explícita, o una solicitud de desconexión implícita como resultado de la finalización de una aplicación, es posible que la lógica de manejo de salidas tenga que limpiar el entorno

de ejecución de salida después de invocar la función de terminación de salida (MQ_TERM_EXIT), si está registrada.

La limpieza del entorno de ejecución de salida implica la liberación de almacenamiento para estructuras de parámetros de salida, posiblemente la supresión de los módulos cargados previamente en la memoria.

Si se producen errores durante esta fase, una llamada MQDISC explícita falla con CompCode MQCC_FAILED y el siguiente código de razón (los errores no se resaltan en las solicitudes de desconexión implícitas):

MQRC_API_EXIT_TERM_ERROR
: Se ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API. La salida no debe devolver ningún error de MQDISC antes o después de las llamadas a la función de salida de la API MQ_TERM*.

Salidas de API en clientes

Un cliente utiliza la salida PreConnect para modificar el comportamiento de las llamadas MQCONN y MQCONNX y no da soporte a las propiedades de salida de API.

Salida PreConnect

En un cliente, la salida PreConnect se puede utilizar para buscar la definición de canal desde un repositorio central, como un servidor LDAP.

La salida PreConnect también puede modificar cualquier parámetro, o todos los parámetros, en una llamada MQCONN o MQCONNX, por ejemplo, el nombre del gestor de colas.

En el caso de las aplicaciones cliente, se debe llamar a la salida PreConnect antes de la salida de API porque la salida de API MQCONN o MQCONNX sólo se llama una vez que se conoce el nombre del gestor de colas y la salida PreConnect puede cambiar este nombre.

Tenga en cuenta que sólo la llamada MQCONN o MQCONNX inicial invoca la salida.

Propiedades de salida de API

En un servidor, las salidas de API pueden registrar una estructura MQXEPO en el momento de la inicialización. La estructura MQXEPO contiene el campo ExitProperties que detalla el grupo de propiedades en las que está interesada la salida. Esto tiene el efecto de generar un descriptor de contexto de propiedad de mensaje independiente que la salida puede manipular por separado de cualquier descriptor de contexto de propiedad de mensaje de aplicación.

En un cliente, las propiedades de salida de API no están soportadas. Si se intenta registrar un nombre de grupo de propiedades en un cliente, la función falla con un código de razón de MQRC_EXIT_PROPS_NOT_SUPPORTED.

Restitución-MQ_BACK_EXIT

MQ_BACK_EXIT proporciona una función de salida de restitución para realizar *antes* y *después* del proceso de restitución. Utilice el identificador de función MQXF_BACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de restitución.

La interfaz para esta función es:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
: Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
: Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_BACK_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Inicio-MQ_BEGIN_EXIT

MQ_BEGIN_EXIT proporciona una función de salida de inicio para realizar *antes* y *después del proceso de llamada* MQBEGIN. Utilice el identificador de función MQXF_BEGIN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQBEGIN.

La interfaz para esta función es:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pBeginOptions (PMQBO)-entrada/salida
Puntero a las opciones de inicio.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        PMQBO    pBeginOptions;  /* Ptr to begin options */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PPMQBO    ppBeginOptions,   /* Address of ptr to begin options */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Devolución de llamada-MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de devolución de llamada. Utilice el identificador de función MQXF_CALLBACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada *before* y *after* .

La interfaz para esta función es:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBContext)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida
Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida
Descriptor de contexto de conexión

pMsgDesc
Descriptor de mensaje

pGetMsgOpts
Opciones que controlan la acción de MQGET

pBuffer
Área para contener los datos del mensaje

pMQCBContext
Datos de contexto para la devolución de llamada

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        PMQMD    pMsgDesc;       /* Message descriptor */
        PMQGMO   pGetMsgOpts;    /* Options that define the operation of the consumer */
        PMQVOID  pBuffer;        /* Area to contain the message data */
        PMQCBC   pContext;       /* Context data for the callback */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP    pExitParms;    /* Exit parameter structure */
PMQAXC    pExitContext;  /* Exit context structure */
PMQHCONN  pHconn;        /* Connection handle */
PPMQMD    ppMsgDesc;     /* Message descriptor */
PPMQGMO   ppGetMsgOpts;  /* Options that define the operation of the consumer */
PPMQVOID  ppBuffer;      /* Area to contain the message data */
PPMQCBC   ppContext;)    /* Context data for the callback */
```

Notas de uso

1. La salida de devolución de llamada se invoca antes de que se invoque al consumidor y después de que se haya completado la función de consumidor del consumidor. Aunque las estructuras MQMD y MQGMO son modificables, el cambio de los valores de la salida anterior no reconduce la recuperación de un mensaje de la cola, ya que el mensaje ya se ha eliminado de la cola para entregarlo a la función de consumidor.

Gestionar funciones de devolución de llamada-MQ_CB_EXIT

MQ_CB_EXIT proporciona una función de salida para realizar *antes* y *después* de la llamada MQCB. Utilice el identificador de función MQXF_CB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQCB.

La interfaz para esta función es:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida
Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida
Descriptor de contexto de conexión

Operación (MQLONG)-entrada/salida
Valor de operación

pCallbackDesc (PMQCBD)-entrada/salida
Descriptor de devolución de llamada

Hobj (MQHOBJ)-entrada/salida
Descriptor de contexto del objeto

pMsgDesc (PMQMD)-entrada/salida
Descriptor de mensaje

pGetMsgOpts (PMQGMO)-entrada/salida
Opciones que controlan la acción de MQCB

CompCode (MQLONG)-entrada/salida
Código de terminación

Razón (MQLONG)-entrada/salida
Código de razón que califica a CompCode

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation value. */
MQCBD    pMsgDesc;       /* Callback descriptor. */
MQHOBJ   Hobj;           /* Object handle. */
PMQMD    pMsgDesc;       /* Message descriptor */
PMQGMO   pGetMsgOpts;    /* Options that define the operation of the consumer */
PMQLONG  CompCode;       /* Completion code. */
PMQLONG  Reason;         /* Reason code qualifying CompCode. */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CB_EXIT (
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
```

```
PMQLONG    pOperation;     /* Callback operation */
PMQHOBJ    pHobj;          /* Object handle */
PPMQMD     ppMsgDesc;      /* Message descriptor */
PPMQGMO    ppGetMsgOpts;   /* Options that control the action of MQCB */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Cerrar-MQ_CLOSE_EXIT

MQ_CLOSE_EXIT proporciona una función de salida de cierre para realizar *antes* y *después del proceso de la llamada MQCLOSE de* . Utilice el identificador de función MQXF_CLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQCLOSE.

La interfaz para esta función es:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pHobj (PMQHOBJ)-entrada
Puntero al descriptor de contexto de objeto.

Opciones (MQLONG)-entrada/salida
Opciones de cierre.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
PMQHOBJ         pHobj;          /* Ptr to object handle */
MQLONG          Options;        /* Close options */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CLOSE_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQHOBJ        ppHobj,         /* Address of ptr to object handle */
PMQLONG         pOptions,       /* Address of close options */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Confirmar-MQ_CMIT_EXIT

MQ_CMIT_EXIT proporciona una función de salida de confirmación para realizar *antes* y *después* del proceso de confirmación. Utilice el identificador de función MQXF_CMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de confirmación.

Si una operación de confirmación falla y la transacción se restituye, la llamada MQCMIT falla con MQCC_WARNING y MQRC_BACKED_OUT. Estos códigos de retorno y de razón se pasan a cualquier *después* de las funciones de salida MQCMIT para dar a las funciones de salida una indicación de que la unidad de trabajo se ha restituido.

La interfaz para esta función es:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CMIT_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida “MQXF_DATA_CONV_ON_GET” en la página 1633 .

   Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

   Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Extensión de conexión y conexión-MQ_CONNX_EXIT

MQ_CONNX_EXIT proporciona una función de salida de conexión para realizar *antes* y *después* del proceso MQCONN, y una función de salida de extensión de conexión para realizar *antes* y *después* del proceso MQCONNX.

Se invoca la misma interfaz, tal como se describe aquí, para las funciones de salida de llamada MQCONN y MQCONNX.

Cuando el agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede conectarse y realizar una serie de llamadas de API de IBM MQ antes de que se conozca completamente el estado del cliente. Estas llamadas de API llaman a las funciones de salida de API con MQAXC basadas en el propio programa MCA (por ejemplo, en los campos UserId y ConnectionName de MQAXC).

Cuando el MCA responde a las llamadas de API de cliente de entrada posteriores, la estructura MQAXC se basa en el cliente de entrada, estableciendo los campos UserId y ConnectionName de forma adecuada.

El nombre del gestor de colas establecido por la aplicación en una llamada MQCONN o MQCONNX se pasa a la llamada de conexión subyacente. Cualquier intento de un *anterior a* MQ_CONNX_EXIT de cambiar el nombre del gestor de colas no tiene ningún efecto.

Utilice los identificadores de función MQXF_CONN y MQXF_CONNX con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de las funciones de salida de llamada MQCONN y MQCONNX de* .

Una salida MQ_CONNX_EXIT llamada por la razón MQXR_BEFORE *no debe* emitir ninguna llamada de API IBM MQ , ya que el entorno correcto no se ha configurado en este momento.

Un MQ_CONNX_EXIT no puede llamar a MQDISC desde una llamada de salida de API para la conexión para la que se está llamando. Esta restricción es aplicable a las salidas de API de cliente y servidor.

La interfaz con MQCONN y MQCONNX es idéntica:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
&pHconn, &CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

pQMgrNombre (PMQCHAR)-entrada
Puntero al nombre de gestor de colas proporcionado en la llamada MQCONNX. La salida no debe cambiar este nombre en la llamada MQCONN o MQCONNX.

pConnectOpts (PMQCNO)-entrada/salida
Puntero a las opciones que controlan la acción de la llamada MQCONNX.

Consulte “MQCNO - Opciones de conexión” en la página 316 para obtener los detalles.

Para la función de salida MQXF_CONN, pConnectOpts apunta a la estructura de opciones de conexión predeterminada (MQCNO_DEFAULT).

pHconn (PMQHCONN)-entrada
Puntero al descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
PMQCHAR         pQMgrName;      /* Ptr to Queue manager name */
PMQCNO          pConnectOpts;   /* Ptr to Connection options */
PMQHCONN        pHconn;         /* Ptr to Connection handle */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOps,
               &pHconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CONNX_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PPMQCHAR        ppQMgrName,     /* Address of ptr to queue manager name */
PPMQCNO         ppConnectOpts,  /* Address of ptr to connection options */
PPMQHCONN       ppHconn,        /* Address of ptr to connection handle */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Notas de uso

1. La interfaz de función MQ_CONNX_EXIT descrita aquí se utiliza tanto para la llamada MQCONN como para la llamada MQCONNX. Sin embargo, se definen puntos de entrada separados para estas dos llamadas. Para interceptar *ambas* llamadas, la llamada MQXEP debe utilizarse al menos dos veces, una con el identificador de función MQXF_CONN y otra con MQXF_CONNX.

   Puesto que la interfaz MQ_CONNX_EXIT es la misma para MQCONN y MQCONNX, se puede utilizar una única función de salida para ambas llamadas; el campo *Function* de la estructura MQAXP indica qué llamada está en curso. De forma alternativa, la llamada MQXEP se puede utilizar para registrar distintas funciones de salida para las dos llamadas.
2. Cuando un agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede emitir una serie de llamadas MQ antes de que se conozca completamente el estado del cliente. Estas llamadas MQ hacen que las funciones de salida de API se invoquen con la estructura MQAXC que contiene datos relacionados con el MCA, y no con el cliente (por ejemplo, el identificador de usuario y el nombre de conexión). Sin embargo, una vez que el estado del cliente es totalmente conocido, las llamadas MQ posteriores dan como resultado que se invoquen las funciones de salida de API con los datos de cliente adecuados en la estructura MQAXC.
3. Todas las funciones de salida MQXR_BEFORE se invocan antes de que el gestor de colas realice cualquier validación de parámetro. Por lo tanto, es posible que los parámetros no sean válidos (incluidos los punteros no válidos para las direcciones de los parámetros).

   La función MQ_CONNX_EXIT se invoca antes de que el gestor de colas realice comprobaciones de autorización.
4. La función de salida no debe cambiar el nombre del gestor de colas especificado en la llamada MQCONN o MQCONNX. Si la función de salida cambia el nombre, los resultados no están definidos.
5. Una función de salida MQXR_BEFORE para MQ_CONNX_EXIT no puede emitir llamadas de MQ distintas de MQXEP.

Devolución de llamada de control-MQ_CTL_EXIT

MQ_CTL_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de devolución de llamada de control. Utilice el identificador de función MQXF_CTL con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de retorno de control.

La interfaz para esta función es:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

donde los parámetros son:

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

Entrada/salida de operación (MQLONG)
La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado

Entrada/salida de ControlOpts (MQCTLO)
Opciones que controlan la acción de MQCTL

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCTLO   ControlOpts;    /* Options that control the action of MQCTL  */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CTL_EXIT (
PMQHCONN  pHconn;          /* Address of connection handle */
PMQLONG   pOperation;      /* Address of operation being processed */
PMQCTLO   pControlOpts;    /* Address of options that control the action of MQCTL  */
PMQLONG   pCompCode;       /* Address of completion code */
PMQLONG   pReason;)        /* Address of reason code qualifying completion code */
```

Desconectar-MQ_DISC_EXIT

MQ_DISC_EXIT proporciona una función de salida de desconexión para realizar *antes* y *después* del proceso de salida MQDISC. Utilice el identificador de función MQXF_DISC con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQDISC.

La interfaz para esta función es

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
&CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

pHconn (PMQHCONN)-entrada
Puntero al descriptor de conexión.

Para la llamada MQDISC anterior, el valor de este campo es uno de los siguientes:

- El descriptor de conexión devuelto en la llamada MQCONN o MQCONNX
- Cero, para entornos en los que un adaptador específico del entorno se ha conectado al gestor de colas
- Un valor establecido por una invocación de función de salida anterior

Para la llamada MQDISC posterior, el valor de este campo es cero o un valor establecido por una invocación de función de salida anterior.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Terminación parcial

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        PMQHCONN        pHconn;         /* Ptr to Connection handle */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_DISC_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PPMQHCONN       ppHconn,        /* Address of ptr to connection handle */
PMQLONG         pCompCode,      /* Address of completion code */
```

```
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Obtener-MQ_GET_EXIT

MQ_GET_EXIT proporciona una función de obtención de salida para realizar *antes* y *después del proceso de la llamada MQGET de* .

Hay dos identificadores de función:

1. Utilice MQXF_GET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQGET.
2. Consulte "MQXF_DATA_CONV_ON_GET" en la página 1633 para obtener información sobre cómo utilizar el identificador de función MQXF_DATA_CONV_ON_GET.

La interfaz para esta función es:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada/salida
El manejador del objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pGetMsgOpts (PMQGMO)-entrada/salida
Puntero para obtener opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

pDataLength (PMQLONG)-entrada/salida
Puntero al campo de longitud de datos.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
MQHOBJ          Hobj;           /* Object handle */
PMQMD           pMsgDesc;       /* Ptr to message descriptor */
PMQPMO          pGetMsgOpts;    /* Ptr to get message options */
MQLONG          BufferLength;   /* Message buffer length */
PMQBYTE         pBuffer;        /* Ptr to message buffer */
PMQLONG         pDataLength;    /* Ptr to data length field */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_GET_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PMQHOBJ         pHobj,          /* Address of object handle */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQGMO         ppGetMsgOpts,   /* Address of ptr to get message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PPMQLONG        ppDataLength,   /* Address of ptr to data length field */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida “MQXF_DATA_CONV_ON_GET” en la página 1633 .

   Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

   Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

MQXF_DATA_CONV_ON_GET

El identificador de función MQXF_DATA_CONV_ON_GET se utiliza con MQ_GET_EXIT.

Consulte MQ_GET_EXIT para obtener información sobre la interfaz para esta llamada y una declaración de lenguaje C de ejemplo.

Notas de uso

Si está registrado, se llama a este punto de entrada cuando llegan mensajes a la aplicación pero antes de que se haya producido cualquier conversión de datos. Esto puede ser útil si la salida de

API necesita realizar el proceso, como el descifrado o la descompresión, antes de que el mensaje se pase a la conversión de datos. La salida puede, si es necesario, hacer que se elude la conversión de datos devolviendo MQXCC_SUPPRESS_FUNCTION; para obtener más información, consulte la estructura MQAXP .

El registro para este punto de entrada en un cliente tiene el efecto de hacer que la conversión de datos se realice localmente en la máquina cliente. Por lo tanto, para que el funcionamiento sea correcto, es posible que sea necesario instalar las salidas de conversión de aplicaciones en el cliente. Tenga en cuenta que MQXF_DATA_CONV_ON_GET también se utiliza para el consumo asíncrono.

Cuando utilice la llamada MQ_GET_EXIT, utilice MQXF_DATA_CONV_ON_GET, con la razón de salida MQXR_BEFORE, para registrar una función de salida de conversión de datos MQGET *anterior a* .

No hay ninguna función de salida MQXR_AFTER para MQXF_DATA_CONV_ON_GET; la función de salida MQXR_AFTER para MQXF_GET proporciona la capacidad necesaria para el proceso de salida después de la conversión de datos.

Se definen puntos de entrada separados para la llamada MQ_GET_EXIT, por lo que para interceptar *ambas* funciones de salida, la llamada MQXEP debe utilizarse dos veces; para esta llamada, utilice el identificador de función MQXF_DATA_CONV_ON_GET.

Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Inicialización-MQ_INIT_EXIT

MQ_INIT_EXIT proporciona inicialización de nivel de conexión, indicada estableciendo ExitReason en MQAXP en MQXR_CONNECTION.

Durante la inicialización, tenga en cuenta lo siguiente:

- La función MQ_INIT_EXIT llama a MQXEP para registrar los verbos de la API IBM MQ y los puntos ENTRY y EXIT en los que está interesado.
- No es necesario que las salidas intercepten todos los verbos de la API de IBM MQ . Las funciones de salida sólo se invocan si se ha registrado un interés.
- El almacenamiento que va a utilizar la salida se puede adquirir al inicializarlo.
- Si falla una llamada a esta función, la llamada MQCONN o MQCONNX que la ha invocado también falla con un CompCode y una razón que dependen del valor del campo ExitResponse en MQAXP.
- Una salida MQ_INIT_EXIT no debe emitir llamadas de API de IBM MQ , porque el entorno correcto no se ha configurado en este momento.
- Si un MQ_INIT_EXIT falla con MQXCC_FAILED, el gestor de colas vuelve de la llamada MQCONN o MQCONNX que le ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.
- Si el gestor de colas encuentra un error al inicializar el entorno de ejecución de la función de salida de API antes de invocar la primera MQ_INIT_EXIT, el gestor de colas devuelve la llamada MQCONN o MQCONNX que ha invocado MQ_INIT_EXIT con MQCC_FAILED y MQRC_API_EXIT_INIT_ERROR.

La interfaz con MQ_INIT_EXIT es:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida
Puntero a código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Puntero al código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_INIT_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Notas de uso

1. La función MQ_INIT_EXIT puede emitir la llamada MQXEP para registrar las direcciones de las funciones de salida para las llamadas MQ concretas que se van a interceptar. No es necesario interceptar todas las llamadas de MQ o interceptar las llamadas MQXR_BEFORE y MQXR_AFTER. Por ejemplo, una suite de salida podría elegir interceptar sólo la llamada MQXR_BEFORE de MQPUT.
2. El almacenamiento que deben utilizar las funciones de salida en la suite de salida puede ser adquirido por la función MQ_INIT_EXIT. De forma alternativa, las funciones de salida pueden adquirir almacenamiento cuando se invocan, como y cuando sea necesario. Sin embargo, todo el almacenamiento debe liberarse antes de que termine la suite de salida; la función MQ_TERM_EXIT puede liberar el almacenamiento o una función de salida invocada anteriormente.
3. Si MQ_INIT_EXIT devuelve MQXCC_FAILED en el campo `ExitResponse` de MQAXP, o falla de alguna otra forma, la llamada MQCONN o MQCONNX que ha hecho que se invoque MQ_INIT_EXIT también falla, con los parámetros **CompCode** y **Reason** establecidos en los valores adecuados.
4. Una función MQ_INIT_EXIT no puede emitir llamadas MQ que no sean MQXEP.

Consultar-MQ_INQ_EXIT

MQ_INQ_EXIT proporciona una función de salida de consulta para realizar *antes* y *después del proceso de la llamada MQINQ de* . Utilice el identificador de función MQXF_INQ con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQINQ.

La interfaz para esta función es:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada
El manejador del objeto.

SelectorCount (MQLONG)-entrada
Recuento de selectores

pSelectors (PMQLONG)-entrada/salida
Puntero a matriz de valores de selector.

IntAttrCount (MQLONG)-entrada
Recuento de atributos enteros.

pIntAttrs (PMQLONG)-entrada/salida
Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG)-entrada/salida
Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR)-entrada/salida
Puntero a matriz de atributos de carácter.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;       /* Exit parameter structure */
        MQAXC    ExitContext;     /* Exit context structure */
        MQHCONN  Hconn;           /* Connection handle */
        MQHOBJ   Hobj;            /* Object handle */
        MQLONG   SelectorCount;   /* Count of selectors */
        PMQLONG  pSelectors;      /* Ptr to array of attribute selectors */
        MQLONG   IntAttrCount;    /* Count of integer attributes */
        PMQLONG  pIntAttrs;       /* Ptr to array of integer attributes */
        MQLONG   CharAttrLength;  /* Length of char attributes array */
        PMQCHAR  pCharAttrs;      /* Ptr to character attributes */
        MQLONG   CompCode;        /* Completion code */
        MQLONG   Reason;          /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_INQ_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQHOBJ   pHobj,            /* Address of object handle */
PMQLONG   pSelectorCount,   /* Address of selector count */
PPMQLONG  ppSelectors,      /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;    /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,       /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,  /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,      /* Address of ptr to character attributes array */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Abrir-MQ_OPEN_EXIT

MQ_OPEN_EXIT proporciona una función de salida abierta para realizar *antes* y *después* del proceso de llamada MQOPEN. Utilice el identificador de función MQXF_OPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQOPEN.

La interfaz para esta función es

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
&pHobj, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pObjDesc (PMQOD)-entrada/salida
Puntero al descriptor de objeto.

Opciones (MQLONG)-entrada/salida
Opciones de apertura.

pHobj (PMQHOBJ)-entrada
Puntero al descriptor de contexto de objeto.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Terminación parcial

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQHCONN         Hconn;          /* Connection handle */
        PMQOD           pObjDesc;       /* Ptr to object descriptor */
        MQLONG          Options;        /* Open options */
        PMQHOBJ         pHobj;          /* Ptr to object handle */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_OPEN_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQOD          ppObjDesc,      /* Address of ptr to object descriptor */
PMQLONG         pOptions,       /* Address of open options */
PPMQHOBJ        ppHobj,         /* Address of ptr to object handle */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Put-MQ_PUT_EXIT

MQ_PUT_EXIT proporciona una función de salida de transferencia para realizar *antes* y *después* del proceso de llamada MQPUT. Utilice el identificador de función MQXF_PUT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQPUT.

La interfaz para esta función es:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada/salida
El manejador del objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO)-entrada/salida
Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
MQHOBJ          Hobj;           /* Object handle */
PMQMD           pMsgDesc;       /* Ptr to message descriptor */
PMQPMO          pPutMsgOpts;    /* Ptr to put message options */
MQLONG          BufferLength;   /* Message buffer length */
PMQBYTE         pBuffer;        /* Ptr to message data */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_PUT_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PMQHOBJ         pHobj,          /* Address of object handle */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQPMO         ppPutMsgOpts,   /* Address of ptr to put message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Notas de uso

- Los mensajes de informe generados por el gestor de colas se saltan el proceso de llamada normal. Como resultado, la función MQ_PUT_EXIT o la función MQPUT1 no pueden interceptar dichos mensajes. Sin embargo, los mensajes de informe generados por el agente de canal de mensajes se procesan normalmente y, por lo tanto, pueden ser interceptados por la función MQ_PUT_EXIT o la función MQ_PUT1_EXIT . Para asegurarse de interceptar todos los mensajes de informe generados por el MCA, deben utilizarse MQ_PUT_EXIT y MQ_PUT1_EXIT .

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT proporciona una función de salida *put one message only* para realizar *antes* y *después del proceso de llamada* MQPUT1 . Utilice el identificador de función MQXF_PUT1 con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de las funciones de salida de llamada* MQPUT1 .

La interfaz para esta función es:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pObjDesc (PMQOD)-entrada/salida
Puntero al descriptor de objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO)-entrada/salida
Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQHCONN         Hconn;          /* Connection handle */
        PMQOD           pObjDesc;       /* Ptr to object descriptor */
        PMQMD           pMsgDesc;       /* Ptr to message descriptor */
        PMQPMO          pPutMsgOpts;    /* Ptr to put message options */
        MQLONG          BufferLength;   /* Message buffer length */
        PMQBYTE         pBuffer;        /* Ptr to message data */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_PUT1_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQOD          ppObjDesc,      /* Address of ptr to object descriptor */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQPMO         ppPutMsgOpts,   /* Address of ptr to put message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Establecer-MQ_SET_EXIT

MQ_SET_EXIT proporciona una función de salida de conjunto para realizar *antes* y *después del proceso de la llamada MQSET de* . Utilice el identificador de función MQXF_SET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQSET.

La interfaz para esta función es:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada
El manejador del objeto.

SelectorCount (MQLONG)-entrada
Recuento de selectores

pSelectors (PMQLONG)-entrada/salida
Puntero a matriz de valores de selector.

IntAttrCount (MQLONG)-entrada
Recuento de atributos enteros.

pIntAttrs (PMQLONG)-entrada/salida
Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG)-entrada/salida
Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR)-entrada/salida
Puntero a valores de atributo de carácter.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;       /* Exit parameter structure */
MQAXC    ExitContext;     /* Exit context structure */
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
PMQLONG  pSelectors;      /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
```

```
        PMQLONG  pIntAttrs;       /* Ptr to array of integer attributes */
        MQLONG   CharAttrLength;  /* Length of char attributes array */
        PMQCHAR  pCharAttrs;      /* Ptr to character attributes */
        MQLONG   CompCode;        /* Completion code */
        MQLONG   Reason;          /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_SET_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQHOBJ   pHobj,            /* Address of object handle */
PMQLONG   pSelectorCount,   /* Address of selector count */
PPMQLONG  ppSelectors,      /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;    /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,       /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,  /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,      /* Address of ptr to character attributes array */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Estado-MQ_STAT_EXIT

MQ_STAT_EXIT proporciona una función de salida de estado para realizar *antes* y *después* del proceso de llamada MQSTAT. Utilice el identificador de función MQXF_STAT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQSTAT.

La interfaz para esta función es:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Tipo (MQLONG)-entrada
Tipo de información de estado a recuperar.

pStatus (PMQSTS)-salida
Puntero al almacenamiento intermedio de estado.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_STAT_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pType             /* Address of status type */
PPMQSTS   ppStatus          /* Address of status buffer */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Terminación-MQ_TERM_EXIT

MQ_TERM_EXIT proporciona terminación a nivel de conexión, registrada con un identificador de función de MQXF_TERM y ExitReason MQXR_CONNECTION. Si está registrado, se llama a MQ_TERM_EXIT una vez para cada solicitud de desconexión.

Como parte de la terminación, el almacenamiento que ya no necesita la salida se puede liberar y se puede realizar cualquier limpieza necesaria.

Si un MQ_TERM_EXIT falla con MQXCC_FAILED, el gestor de colas devuelve el MQDISC que lo ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si el gestor de colas encuentra un error al terminar el entorno de ejecución de la función de salida de API después de invocar el último MQ_TERM_EXIT, el gestor de colas devuelve la llamada MQDISC que ha invocado MQ_TERM_EXIT con MQCC_FAILED y MQRC_API_EXIT_TERM_ERROR

La interfaz para esta función es:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_TERM_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Notas de uso

1. La función MQ_TERM_EXIT es opcional. No es necesario que una suite de salida registre una salida de terminación si no hay que realizar ningún proceso de terminación.

   Si las funciones pertenecientes a la suite de salida adquieren recursos durante la conexión, una función MQ_TERM_EXIT es un punto conveniente en el que liberar esos recursos, por ejemplo, liberando almacenamiento obtenido dinámicamente.
2. Si se registra una función MQ_TERM_EXIT cuando se emite la llamada MQDISC, la función de salida se invoca después de que se hayan invocado todas las funciones de salida MQDISC.
3. Si MQ_TERM_EXIT devuelve MQXCC_FAILED en el campo `ExitResponse` de MQAXP, o falla de alguna otra forma, la llamada MQDISC que ha hecho que se invoque MQ_TERM_EXIT también falla, con los parámetros **CompCode** y **Reason** establecidos en los valores adecuados.

Registrar suscripción-MQ_SUB_EXIT

MQ_SUB_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de reregistro de suscripción. Utilice el identificador de función MQXF_SUB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de registrationcall de suscripción.

La interfaz para esta función es:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

pSubDesc-entrada/salida
Matriz de selectores de atributos.

pHobj -entrada/salida
Descriptor de contexto del objeto

pHsub (MQHOBJ) entrada/salida
Manejador de suscripciones

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
PMQSD    pSubDesc;       /* Subscription descriptor */
PMQHOBJ  pHobj;          /* Object Handle */
PMQHOBJ  pHsub;          /* Subscription handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
PMQAXP    pExitParms;      /* Exit parameter structure */
PMQAXC    pExitContext;    /* Exit context structure */
PMQHCONN  pHconn;          /* Connection handle */
PPMQSD    ppSubDesc;       /* Subscription descriptor */
PPMQHOBJ  ppHobj;          /* Object Handle */
PPMQHOBJ  ppHsub;          /* Subscription handle */
PMQLONG   pCompCode;       /* Completion code */
PMQLONG   pReason;         /* Reason code qualifying completion code */
```

Solicitud de suscripción-MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de solicitud de suscripción. Utilice el identificador de función MQXF_SUBRQ con las

razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada de solicitud de suscripción *before* y *after* .

La interfaz para esta función es:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

pHsub (MQHOBJ) entrada/salida
Manejador de suscripciones

Entrada/salida de acción (MQLONG)
Acción

pSubRqOpts (MQSRO) entrada/salida

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        PMQLONG  pHsub;          /* Subscription handle */
        MQLONG   Action;         /* Action */
        PMQSRO   pSubRqOpts;     /* Subscription Request Options */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_SUBRQ_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PPMQHOBJ  ppHsub;           /* Address of Subscription handle */
PMQLONG   pAction;          /* Address of Action */
PPMQSRO   ppSubRqOpts;      /* Address of Subscription Request Options */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

xa_close-XA_CLOSE_EXIT

XA_CLOSE_EXIT proporciona una función de salida xa_close para realizar antes y después del proceso xa_close. Utilice el identificador de función MQXF_XACLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_close.

La interfaz para esta función es:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXa_info (PMQCHAR)-entrada/salida
Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP    ExitParms;    /* Exit parameter structure */
     MQAXC    ExitContext;  /* Exit context structure */
     MQHCONN  Hconn;        /* Connection handle */
     PMQCHAR  pXa_info;     /* Instance-specific RM info */
     MQLONG   Rmid;         /* Resource manager identifier */
     MQLONG   Flags;        /* Resource manager options*/
     MQLONG   XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_CLOSE_EXIT (
     PMQAXP    pExitParms,    /* Address of exit parameter structure */
     PMQAXC    pExitContext,  /* Address of exit context structure */
     PMQHCONN  pHconn,        /* Address of connection handle */
     PPMQCHAR  ppXa_info,     /* Address of instance-specific RM info */
```

```
     PMQLONG   pRmid,         /* Address of resource manager identifier */
     PMQLONG   pFlags,        /* Address of resource manager options*/
     PMQLONG   pXARetCode);   /* Address of response from XA call */
```

xa_commit-XA_COMMIT_EXIT

XA_COMMIT_EXIT proporciona una función de salida xa_commit para realizar antes y después del proceso xa_commit. Utilice el identificador de función MQXF_XACOMMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_commit.

La interfaz para esta función es:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP    ExitParms;   /* Exit parameter structure */
     MQAXC    ExitContext; /* Exit context structure */
     MQHCONN  Hconn;       /* Connection handle */
     MQPTR    pXID;        /* Transaction branch ID */
     MQLONG   Rmid;        /* Resource manager identifier */
     MQLONG   Flags;       /* Resource manager options*/
     MQLONG   XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_COMMIT_EXIT (
     PMQAXP    pExitParms,   /* Address of exit parameter structure */
     PMQAXC    pExitContext, /* Address of exit context structure */
     PMQHCONN  pHconn,       /* Address of connection handle */
     PMQPTR    ppXID,        /* Address of transaction branch ID */
     PMQLONG   pRmid,        /* Address of resource manager identifier */
     PMQLONG   pFlags,       /* Address of resource manager options*/
     PMQLONG   pXARetCode);  /* Address of response from XA call */
```

xa_complete-XA_COMPLETE_EXIT

XA_COMPLETE_EXIT proporciona una función de salida xa_complete para realizar antes y después del proceso xa_complete. Utilice el identificador de función MQXF_XACOMPLETE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_complete.

La interfaz para esta función es:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pHandle (PMQLONG)-entrada/salida
Puntero a operación asíncrona.

pRetVal (PMQLONG)-entrada/salida
Valor de retorno de la operación asíncrona.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     PMQLONG pHandle;     /* Ptr to asynchronous op */
     PMQLONG pRetval;     /* Return value of async op */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_COMPLETE_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PPMQLONG ppHandle,     /* Address of ptr to asynchronous op */
     PPMQLONG ppRetval,     /* Address of return value of async op */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_end-XA_END_EXIT

XA_END_EXIT proporciona una función de salida xa_end para realizar antes y después del proceso xa_end. Utilice el identificador de función MQXF_XAEND con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_end.

La interfaz para esta función es:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_END_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_forget-XA_OLVIDO_SALIDA

XA_OLVI_EXIT proporciona una función de salida xa_olvidar para realizar antes y después del proceso xa_olvidar. Utilice el identificador de función MQXF_XAFORGET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_forget.

La interfaz para esta función es:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_FORGET_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_open-XA_OPEN_EXIT

XA_OPEN_EXIT proporciona una función de salida xa_open para realizar antes y después del proceso xa_open. Utilice el identificador de función MQXF_XAOPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_open.

La interfaz para esta función es:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXa_info (PMQCHAR)-entrada/salida
Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     PMQCHAR pXa_info;    /* Instance-specific RM info */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_OPEN_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PPMQCHAR ppXa_info,    /* Address of instance-specific RM info */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_prepare-XA_PREPARE_EXIT

XA_PREPARE_EXIT proporciona una función de salida xa_prepare para realizar antes y después del proceso xa_prepare. Utilice el identificador de función MQXF_XAPREPARE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_prepare.

La interfaz para esta función es:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms;  /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_PREPARE_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_recover-XA_RECOVER_EXIT

XA_RECOVER_EXIT proporciona una función de salida xa_recover para realizar antes y después del proceso xa_recover. Utilice el identificador de función MQXF_XARECOVER con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_recover.

La interfaz para esta función es:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Recuento (MQLONG)-entrada/salida
Máximo de XID en matriz XID

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Count;       /* Max XIDs in XID array */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_RECOVER_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pCount,       /* Address of max XIDs in XID array */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_rollback-XA_ROLLBACK_EXIT

XA_ROLLBACK_EXIT proporciona una función de salida xa_rollback para realizar antes y después del proceso xa_rollback. Utilice el identificador de función MQXF_XAROLLBACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_rollback.

La interfaz para esta función es:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_ROLLBACK_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_start-XA_START_EXIT

XA_START_EXIT proporciona una función de salida xa_start para realizar antes y después del proceso xa_start. Utilice el identificador de función MQXF_XASTART con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_start.

La interfaz para esta función es:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_START_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

ax_reg-AX_REG_EXIT

AX_REG_EXIT proporciona una función de salida ax_reg para realizar antes y después del proceso ax_reg. Utilice el identificador de función MQXF_AXREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_reg.

La interfaz para esta función es:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP  ExitParms;   /* Exit parameter structure */
     MQAXC  ExitContext; /* Exit context structure */
     MQPTR  pXID;        /* Transaction branch ID */
```

```
     MQLONG Rmid;        /* Resource manager identifier */
     MQLONG Flags;       /* Resource manager options*/
     MQLONG XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY AX_REG_EXIT (
     PMQAXP  pExitParms,   /* Address of exit parameter structure */
     PMQAXC  pExitContext, /* Address of exit context structure */
     PMQPTR  ppXID,        /* Address of transaction branch ID */
     PMQLONG pRmid,        /* Address of resource manager identifier */
     PMQLONG pFlags,       /* Address of resource manager options*/
     PMQLONG pXARetCode);  /* Address of response from XA call */
```

ax_unreg-AX_UNREG_EXIT

AX_UNREG_EXIT proporciona una función de salida ax_unreg para realizar antes y después del proceso ax_unreg. Utilice el identificador de función MQXF_AXUNREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_unreg.

La interfaz para esta función es:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     MQAXP  ExitParms;   /* Exit parameter structure */
     MQAXC  ExitContext; /* Exit context structure */
     MQLONG Rmid;        /* Resource manager identifier */
     MQLONG Flags;       /* Resource manager options*/
     MQLONG XARetCode;   /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY AX_UNREG_EXIT (
     PMQAXP  pExitParms,   /* Address of exit parameter structure */
     PMQAXC  pExitContext, /* Address of exit context structure */
     PMQLONG pRmid,        /* Address of resource manager identifier */
```

```
     PMQLONG pFlags,       /* Address of resource manager options*/
     PMQLONG pXARetCode);  /* Address of response from XA call */
```

Información general sobre la invocación de funciones de salida

Este tema proporciona algunas directrices generales para ayudarle a planificar las salidas, especialmente relacionadas con el manejo de errores y sucesos inesperados.

Error de salida

Si una función de salida termina de forma anómala después de una llamada MQGET destructiva, fuera de punto de sincronismo, pero antes de que el mensaje se haya pasado a la aplicación, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación.

En este caso, es posible que se pierda el mensaje. Esto es como lo que sucede cuando una aplicación falla inmediatamente después de recibir un mensaje de una cola.

La llamada MQGET puede completarse con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si una función de salida de llamada de API *before* termina de forma anómala, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación sin procesar la llamada de API. En este caso, la función de salida debe recuperar los recursos que posee.

Si las salidas encadenadas están en uso, las salidas de llamada de API *after* para cualquier salida de llamada de API *before* que se haya controlado correctamente pueden ser controladas por sí mismas. La llamada de API puede fallar con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Manejo de errores de ejemplo para funciones de salida

El diagrama siguiente muestra los puntos (e *N*) en el que se pueden producir errores. Es sólo un ejemplo para mostrar cómo se comportan las salidas y debe leerse junto con la tabla siguiente. En este ejemplo, se invocan dos funciones de salida antes y después de cada llamada de API para mostrar el comportamiento con salidas encadenadas.

```
Application ErrPt     Exit function            API call
----------- -----     -------------            --------

Start

MQCONN  -->
            e1
                         MQ_INIT_EXIT
            e2
                   before MQ_CONNX_EXIT  1
            e3
                   before MQ_CONNX_EXIT  2
            e4
                                            -->  MQCONN
            e5
                   after  MQ_CONNX_EXIT  2
            e6
                   after  MQ_CONNX_EXIT  1
            e7
        <--
MQOPEN  -->
                   before MQ_OPEN_EXIT  1
            e8
                   before MQ_OPEN_EXIT  2
            e9
                                            -->  MQOPEN
            e10
                   after  MQ_OPEN_EXIT  2
            e11
                   after  MQ_OPEN_EXIT  1
            e12
        <--
MQPUT   -->
                   before MQ_PUT_EXIT   1
            e13
                   before MQ_PUT_EXIT   2
            e14
                                            -->  MQPUT
```

```
            e15
                  after  MQ_PUT_EXIT   2
            e16
                  after  MQ_PUT_EXIT   1
            e17
        <--
MQCLOSE -->
                  before MQ_CLOSE_EXIT 1
            e18
                  before MQ_CLOSE_EXIT 2
            e19
                                          -->  MQCLOSE
            e20
                  after  MQ_CLOSE_EXIT 2
            e21
                  after  MQ_CLOSE_EXIT 1
            e22
        <--
MQDISC  -->
                  before MQ_DISC_EXIT  1
            e23
                  before MQ_DISC_EXIT  2
            e24
                                          -->  MQDISC
            e25
                  after  MQ_DISC_EXIT  2
            e26
                  after  MQ_DISC_EXIT  1
            e27

        <--

end
```

La tabla siguiente lista las acciones que se deben realizar en cada punto de error. Sólo se ha cubierto un subconjunto de los puntos de error, ya que las reglas que se muestran aquí se pueden aplicar a todos los demás. Son las acciones que especifican el comportamiento previsto en cada caso.

Tabla 837. Errores de salida de API y acciones adecuadas a realizar

Err Pt	Descripción	Acciones
e1	Error al configurar el entorno.	1. Deshacer configuración de entorno según sea necesario 2. No hay funciones de salida de unidad 3. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	La función MQ_INIT_EXIT se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Limpiar entorno 2. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_* [1] 2. Limpiar entorno

Tabla 837. Errores de salida de API y acciones adecuadas a realizar (continuación)		
Err Pt	**Descripción**	**Acciones**
e3	*Antes* , la función MQ_CONNX_EXIT 1 finaliza con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad MQ_TERM_EXIT, función 2. Limpiar entorno 3. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_ * [1] 2. Función MQ_TERM_EXIT de unidad, si es necesario 3. Limpiar entorno si es necesario
e4	*Antes* , la función MQ_CONNX_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_CONNX_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Limpiar entorno 4. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_ * [1] 2. Unidad *después* de la función MQ_CONNX_EXIT 1 si la salida no se suprime 3. Función MQ_TERM_EXIT de unidad, si es necesario 4. Limpiar entorno si es necesario
e5	La llamada MQCONN falla.	1. Pasar MQCONN CompCode y Razón 2. Unidad *después* de la función MQ_CONNX_EXIT 2 si *antes* MQ_CONNX_EXIT 2 se ha ejecutado correctamente y la salida no se suprime 3. Unidad *después* de la función MQ_CONNX_EXIT 1 si *antes* MQ_CONNX_EXIT 1 se ha ejecutado correctamente y la salida no se suprime 4. Unidad MQ_TERM_EXIT, función 5. Limpiar entorno
e6	*Después de* la función MQ_CONNX_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_CONNX_EXIT 1 2. Complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_ * [1] 2. Unidad *después de la función* MQ_CONNX_EXIT 1 si es necesario

Tabla 837. Errores de salida de API y acciones adecuadas a realizar (continuación)		
Err Pt	**Descripción**	**Acciones**
e7	*Después de que la función* MQ_CONNX_EXIT 1 se complete con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED, complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ * [1]
e8	*Antes* la función MQ_OPEN_EXIT 1 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED, complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ * [1]
e9	*Antes* , la función MQ_OPEN_EXIT 2 finaliza con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ * [1]
e10	La llamada MQOPEN falla	1. Pasar MQOPEN CompCode y Razón 2. Unidad *después de la función* MQ_OPEN_EXIT 2 si la salida no se suprime 3. Unidad *después de la función* MQ_OPEN_EXIT 1 si la salida no se ha suprimido y si las salidas encadenadas no se han suprimido
e11	*Después de que la función MQ_OPEN_EXIT 2 de* se complete con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_ * [1] 2. Unidad *después de la función* MQ_OPEN_EXIT 1 si no se suprime la salida
e25	*Después de* la función MQ_DISC_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_DISC_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza 4. Completar llamada MQDISC con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actúe como para los valores de MQXCC_ * y MQXR2_ * [1] 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza

Nota:

1. Los valores de MQXCC_ * y MQXR2_* y sus acciones correspondientes se definen en Cómo procesan los gestores de colas las funciones de salida.

Los campos ExitResponse se han establecido incorrectamente

Este tema proporciona información sobre lo que sucedería cuando el campo ExitResponse se establece en cualquier valor que no sean los valores soportados.

Si el campo ExitResponse se establece en un valor distinto de uno de los valores soportados, se aplican las acciones siguientes:

- Para una función de salida de API MQCONN o MQDISC *anterior a* :
  - El valor ExitResponse2 se ignora.
  - No se invocan más *antes* de las funciones de salida en la cadena de salida (si las hay); no se emite la propia llamada de API.
  - Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
  - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de estas funciones de salida.
  - La llamada MQCONN o MQDISC falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API *anterior a* IBM MQ que no sea MQCONN o MQDISC:
  - El valor ExitResponse2 se ignora.
  - No se invocan más *antes* o *después* de las funciones de conversión de datos en la cadena de salida (si las hay).
  - Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
  - La propia llamada de API IBM MQ no se emite.
  - La llamada de API IBM MQ falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API MQCONN o MQDISC *después de* :
  - El valor ExitResponse2 se ignora.
  - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
  - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes* o *después* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de la salida.
  - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
  - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
  - La llamada de API IBM MQ se ha emitido correctamente.
- Para una función de salida de llamada de API *after* IBM MQ que no sea MQCONN o MQDISC:
  - El valor ExitResponse2 se ignora.
  - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
  - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
  - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
  - La llamada de API IBM MQ se ha emitido correctamente.
- Para la conversión de datos *before* al obtener la función de salida:

- El valor ExitResponse2 se ignora.
- Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
- El mensaje no se convierte y el mensaje no convertido se devuelve a la aplicación.
- Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
- Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
- La llamada de API IBM MQ se ha emitido correctamente.

Nota: Dado que el error es con la salida, es mejor devolver MQRC_API_EXIT_ERROR que devolver MQRC_NOT_CONVERT.

Si una función de salida establece el campo ExitResponse2 en un valor distinto de uno de los valores soportados, en su lugar se asume un valor de MQXR2_DEFAULT_CONTINUATION .

Información de consulta sobre la interfaz de servicios instalables

Esta colección de temas proporciona información de referencia para los servicios instalables.

Las funciones y los tipos de datos se listan en orden alfabético dentro del grupo para cada tipo de servicio.

Conceptos relacionados

ULW Servicios y componentes instalables para UNIX, Linux y Windows

IBM i Servicios y componentes instalables para IBM i

Tareas relacionadas

Extensión de recursos del gestor de colas

ULW Configuración de servicios instalables

Referencia relacionada

IBM i Información de referencia de la interfaz de servicios instalables para IBMi

Cómo se muestran las funciones

Cómo se documentan las funciones de servicios instalables.

Para cada función hay una descripción, incluido el identificador de función (para MQZEP).

Los *parámetros* se muestran en el orden en el que deben aparecer. Todos ellos deben estar presentes.

Cada nombre de parámetro va seguido de su tipo de datos. Estos son los tipos de datos elementales descritos en “Tipos de datos elementales” en la página 235.

También se proporciona la invocación del lenguaje C, después de la descripción de los parámetros.

MQZ_AUTHENTICATE_USER-Autenticar usuario

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para autenticar un usuario o para establecer campos de contexto de identidad. Se invoca cuando se establece el contexto de aplicación de usuario IBM MQ .

El contexto de aplicación se establece durante las llamadas de conexión en el punto en el que se inicializa el contexto de usuario de la aplicación y en cada punto en el que se cambia el contexto de usuario de la aplicación. Cada vez que se realiza una llamada de conexión, la información de contexto de usuario de la aplicación se vuelve a adquirir en el campo *IdentityContext* .

El identificador de función para esta función (para MQZEP) es MQZID_AUTHENTICATE_USER.

Sintaxis

MQZ_AUTHENTICATE_USER (*QMgrName* , *SecurityParms* , *ApplicationContext* , *IdentityContext* , *CorrelationPtr* , *ComponentData* , *Continuation* , *CompCode* , *Motivo*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SecurityParms

Tipo: MQCSP-entrada

Parámetros de seguridad. Datos relacionados con el ID de usuario, la contraseña y el tipo de autenticación. Si el atributo AuthenticationType de la estructura MQCSP se especifica como MQCSP_AUTH_USER_ID_AND_PWD, tanto el ID de usuario como la contraseña se comparan con los campos equivalentes del parámetro IdentityContext (MQZIC) para determinar si coinciden. Para obtener más información, consulte "MQCSP-Parámetros de seguridad" en la página 336.

Durante una llamada MQCONN MQI, este parámetro contiene valores nulos o predeterminados.

ApplicationContext

Tipo: MQZAC-entrada

Contexto de aplicación. Datos relacionados con la aplicación de llamada. Consulte MQZAC-Contexto de aplicación para obtener más detalles.

Durante cada llamada MQCONN o MQCONNX MQI, se vuelve a adquirir la información de contexto de usuario de la estructura MQZAC.

IdentityContext

Tipo: MQZIC-entrada/salida

Contexto de identidad. En la entrada de la función de autenticar usuario, identifica el contexto de identidad actual. La función de autenticar usuario puede cambiar esto, momento en el que el gestor de colas adopta el nuevo contexto de identidad. Consulte MQZIC-Contexto de identidad para obtener más detalles sobre la estructura MQZIC.

CorrelationPtr

Tipo: MQPTR-salida

Puntero de correlación. Especifica la dirección de cualquier dato de correlación. Este puntero se pasa posteriormente a otras llamadas de OAM.

ComponentData

Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Distintivo de continuación. Puede especificar los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare los parámetros pasados al servicio como se indica a continuación:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP     SecurityParms;      /* Security parameters */
MQZAC     ApplicationContext; /* Application context */
MQZIC     IdentityContext;    /* Identity context */
MQPTR     CorrelationPtr;     /* Correlation pointer */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY-Comprobar autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

Sintaxis

MQZ_CHECK_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuya autorización para el objeto se va a comprobar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityName. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización
Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

Las autorizaciones siguientes se aplican al uso de las llamadas MQI:

MQZAO_CONNECT
Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE
Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT
Principal. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT
Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE
Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET
Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT
Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ IDENTITY_ CONTEXT
Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ ALL_CONTEXT
Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI
Todas las autorizaciones MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR
Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE
Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY
Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE
Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR
Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR
Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL
Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED
Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN
Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las autorizaciones siguientes se aplican tanto al uso de la MQI como a la administración de un gestor de colas:

MQZAO_TODOS
Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NONE
Sin autorizaciones.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación* .

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
```

```
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY_2 -Comprobar autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 es como MQZ_CHECK_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData** .

Sintaxis

MQZ_CHECK_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad con autorización sobre el objeto que se va a comprobar. Para obtener más detalles, consulte "MQZED-Descriptor de entidad" en la página 1724.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

Las autorizaciones siguientes se aplican al uso de las llamadas MQI:

MQZAO_CONNECT
Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE
Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT
Principal. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT
Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE
Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET
Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT
Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ IDENTITY_ CONTEXT
Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ ALL_CONTEXT
Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI
Todas las autorizaciones MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR
Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE
Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY
Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE
Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR
Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR
Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL
Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED
Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN
Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las autorizaciones siguientes se aplican tanto al uso de la MQI como a la administración de un gestor de colas:

MQZAO_TODOS
Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NONE
Sin autorizaciones.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-Comprobar si el usuario tiene privilegios

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_6 y la invoca el gestor de colas para determinar si un usuario especificado es un usuario privilegiado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_PRIVILEGED.

Sintaxis

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad que se va a comprobar. Para obtener más información, consulte “MQZED-Descriptor de entidad” en la página 1724.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityData. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ComponentData
Tipo: MQBYTExComponentDataLength -entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación* .

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NO_PRIVILEGIADO
(2584, X'A18') Este usuario no es un ID de usuario privilegiado.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                    ComponentData, &Continuation,
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY-Copiar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización. Lo inicia el gestor de colas para copiar todas las autorizaciones que están actualmente en vigor para un objeto de referencia en otro objeto.

El identificador de función para esta función (para MQZEP) es MQZID_COPY_ALL_AUTHORITY.

Sintaxis

MQZ_COPY_ALL_AUTHORITY(*QMgrName* , *RefObjectName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

Nombre de RefObject
Tipo: MQCHAR48 -entrada

Nombre de objeto de referencia. El nombre del objeto de referencia, cuyas autorizaciones se van a copiar. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a establecer los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *RefObjectName* y *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

ComponentData
Tipo: MQBYTExComponentDataLength -entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Objeto de referencia desconocido.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY-Suprimir autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas para suprimir todas las autorizaciones asociadas con el objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_AUTHORITY.

Sintaxis

MQZ_DELETE_AUTHORITY(*QMgrName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a suprimir los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTORITY_DATA-Enumerar datos de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_4 y la inicia repetidamente el gestor de colas para recuperar todos los datos de autorización que coinciden con los criterios de selección especificados en la primera invocación.

El identificador de función para esta función (para MQZEP) es MQZID_ENUMERATE_AUTORITY_DATA.

Sintaxis

MQZ_ENUMERATE_AUTHORITY_DATA(*QMgrName* , *StartEnumeration* , *Filter* , *AuthorityBufferLength* , *AuthorityBuffer* , *AuthorityDataLength* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

StartEnumeration
Tipo: MQLONG - entrada

Distintivo que indica si la llamada puede iniciar la enumeración. Esto indica si la llamada puede iniciar la enumeración de datos de autorización o continuar la enumeración de datos de autorización iniciada por una llamada anterior a MQZ_ENUMERATE_AUTORITY_DATA. El valor es uno de los valores siguientes:

MQZSE_START
Iniciar enumeración. La llamada se inicia con este valor para iniciar la enumeración de datos de autorización. El parámetro **Filter** especifica los criterios de selección que se deben utilizar para seleccionar los datos de autorización devueltos por esta y por llamadas sucesivas.

MQZSE_CONTINUE
Continuar enumeración. La llamada se inicia con este valor para continuar la enumeración de datos de autorización. El parámetro **Filter** se ignora en este caso y se puede especificar como puntero nulo (los criterios de selección se determinan mediante el parámetro **Filter** especificado por la llamada que tenía *StartEnumeration* establecido en MQZSE_START).

Filtro
Tipo: MQZAD-entrada

Filtro. Si *StartEnumeration* es MQZSE_START, *Filter* especifica los criterios de selección que se van a utilizar para seleccionar los datos de autorización que se van a devolver. Si *Filter* es el puntero nulo, no se utiliza ningún criterio de selección, es decir, se devuelven todos los datos de autorización. Consulte "MQZAD-Datos de autorización" en la página 1721 para obtener detalles de los criterios de selección que se pueden utilizar.

Si *StartEnumeration* es MQZSE_CONTINUE, *Filter* se ignora y se puede especificar como puntero nulo.

AuthorityBufferLongitud
Tipo: MQLONG - entrada

Longitud de *AuthorityBuffer*. Es la longitud en bytes del parámetro **AuthorityBuffer** . El almacenamiento intermedio de autorización debe ser lo suficientemente grande para dar cabida a los datos que se van a devolver.

AuthorityBuffer
Tipo: MQZAD-salida

Datos de autorización. Es el almacenamiento intermedio en el que se devuelven los datos de autorización. El almacenamiento intermedio debe ser lo suficientemente grande para acomodar una estructura MQZAD, una estructura MQZED, más el nombre de entidad más largo y el nombre de dominio más largo definido.

Nota: Nota: Este parámetro se define como un MQZAD, ya que el MQZAD siempre se produce al principio del almacenamiento intermedio. Sin embargo, si el almacenamiento intermedio se declara como MQZAD, el almacenamiento intermedio será demasiado pequeño; debe ser mayor que un MQZAD para que pueda acomodar los nombres de entidad y dominio MQZAD, MQZED, más.

AuthorityDataLongitud
Tipo: MQLONG - salida

Longitud de los datos devueltos en *AuthorityBuffer*. Si el almacenamiento intermedio de autorización es demasiado pequeño, *AuthorityDataLength* se establece en la longitud del almacenamiento intermedio necesaria y la llamada devuelve el código de terminación MQCC_FAILED y el código de razón MQRC_BUFFER_LENGTH_ERROR.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_ENUMERATE_AUTORITY_DATA, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') No hay datos disponibles.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
```

```
                                &Continuation, &CompCode,
                                &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;               /* Queue manager name */
MQLONG    StartEnumeration;       /* Flag indicating whether call should
                                     start enumeration */
MQZAD     Filter;                 /* Filter */
MQLONG    AuthorityBufferLength;  /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;        /* Authority data */
MQLONG    AuthorityDataLength;    /* Length of data returned in
                                     AuthorityBuffer */
MQBYTE    ComponentData[n];       /* Component data */
MQLONG    Continuation;           /* Continuation indicator set by
                                     component */
MQLONG    CompCode;               /* Completion code */
MQLONG    Reason;                 /* Reason code qualifying CompCode */
```

MQZ_FREE_USER-Usuario libre

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para liberar el recurso asignado asociado.

Se inicia cuando una aplicación ha terminado de ejecutarse en todos los contextos de usuario, por ejemplo, durante una llamada MQI MQDISC.

El identificador de función para esta función (para MQZEP) es MQZID_FREE_USER.

Sintaxis

MQZ_FREE_USER(*QMgrName* , *FreeParms* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

FreeParms

Tipo: MQZFP-entrada

Parámetros libres. Estructura que contiene datos relacionados con el recurso que se va a liberar. Para obtener más detalles, consulte “MQZFP-Parámetros libres” en la página 1726.

ComponentData

Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Distintivo de continuación. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                    IdentityContext, CorrelationPtr, ComponentData,
                    &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZFP     FreeParms;          /* Resource to be freed */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY-Obtener autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado, incluidas (si la entidad es un principal) las autorizaciones que poseen los grupos de los que el principal es miembro. Las autorizaciones de perfiles genéricos se incluyen en el conjunto de autorizaciones devuelto.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

Sintaxis

MQZ_GET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuyo acceso al objeto se va a recuperar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se va a recuperar el acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TOPIC

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData

Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY_2 -Obtener autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 es como MQZ_GET_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData** .

Sintaxis

MQZ_GET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad para la que se va a recuperar la autorización para el objeto. Para obtener más detalles, consulte “MQZED-Descriptor de entidad” en la página 1724.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Sintaxis

MQZ_GET_AUTHORITY_2 *(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Invocación en C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
```

```
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY-Obtener autorización explícita

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado, incluidas (si la entidad es un principal) las autorizaciones que poseen los grupos de los que el principal es miembro. Las autorizaciones de perfiles genéricos se incluyen en el conjunto de autorizaciones devuelto.

En UNIX, para el gestor de autorizaciones sobre objetos (OAM) de IBM MQ incorporado, la autorización devuelta es la que sólo posee el grupo primario del principal.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName

Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY_2 -Obtener autorización explícita (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que tiene el grupo primario del principal con nombre para acceder a un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 es como MQZ_GET_EXPLICIT_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData** .

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización al objeto se va a recuperar. Para obtener más detalles, consulte "MQZED-Descriptor de entidad" en la página 1724.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY-Inicializar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_AUTHORITY.

Sintaxis

MQZ_INIT_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY
Inicialización primaria.

MQZIO_SECUNDARIO
Inicialización secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

Longitud de ComponentData
Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Versión
Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en la devolución el gestor de colas no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_AUTHORITY del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1
Versión 1.

MQZAS_VERSION_2
Versión 2.

MQZAS_VERSION_3
Versión 3.

MQZAS_VERSION_4
Versión 4.

MQZAS_VERSION_5
Versión 5.

MQZAS_VERSION_6
Versión 6.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') La inicialización ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

MQZ_INQUIRE-Consultar servicio de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para consultar la funcionalidad soportada.

Cuando se utilizan varios componentes de servicio, se llama a los componentes de servicio en orden inverso al orden en el que se instalaron.

El identificador de función para esta función (para MQZEP) es MQZID_INQUIRE.

Sintaxis

MQZ_INQUIRE(*QMgrName* , *SelectorCount* , *Selectors* , *IntAttrCount* , *IntAttrs* , *CharAttrLength* , *CharAttrs* , *SelectorReturned* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SelectorCount
Tipo: MQLONG - entrada

Número de selectores. Número de selectores proporcionados en el parámetro **Selectors** .

El valor debe estar en el rango de 0 a 256.

Selectores
Tipo: MQLONGxSelectorRecuento-entrada

Matriz de selectores. Cada selector identifica un atributo necesario y debe ser uno de los siguientes:

- MQIACF_INTERFACE_VERSION (entero)
- MQIACF_USER_ID_SUPPORT (entero)
- MQCACF_SERVICE_COMPONENT (carácter)

Los selectores se pueden especificar en cualquier orden. El número de selectores de la matriz se indica mediante el parámetro **SelectorCount** .

Los atributos enteros identificados por los selectores se devuelven en el parámetro **IntAttrs** en el mismo orden en que aparecen en *Selectors*.

Los atributos de carácter identificados por los selectores se devuelven en el parámetro **CharAttrs** en el mismo orden en que aparecen *Selectors*.

IntAttrCount
Tipo: MQLONG - entrada

Número de atributos enteros proporcionados en el parámetro IntAttrs .

El valor debe estar en el rango de 0 a 256.

IntAttrs
Tipo: MQLONG x IntAttrRecuento-salida

Atributos enteros. Matriz de atributos enteros. Los atributos de entero se devuelven en el mismo orden que los selectores de entero correspondientes en la matriz *Selectors* .

Recuento de CharAttr
Tipo: MQLONG - entrada

Longitud del almacenamiento intermedio de atributos de caracteres. Longitud en bytes del parámetro **CharAttrs** .

El valor debe ser como mínimo la suma de las longitudes de los atributos de caracteres solicitados. Si no se solicita ningún atributo de carácter, cero es un valor válido.

CharAttrs
Tipo: MQLONG x CharAttrRecuento-salida

Almacenamiento intermedio de atributos de caracteres. Almacenamiento intermedio que contiene atributos de caracteres, concatenados entre sí. Los atributos de carácter se devuelven en el mismo orden que los selectores de caracteres correspondientes en la matriz *Selectors* .

La longitud del almacenamiento intermedio la proporciona el parámetro de recuento CharAttr.

SelectorReturned
Tipo: MQLONG x SelectorCount -entrada

Se ha devuelto el selector. Matriz de valores que identifican qué atributos se han devuelto del conjunto solicitado por los selectores en el parámetro Selectores. El número de valores de esta matriz se indica mediante el parámetro **SelectorCount** . Cada valor en la matriz se relaciona con el selector desde la posición correspondiente en la matriz de selectores. Cada valor es uno de los siguientes:

MQZSL_DEVUELTO
Se ha devuelto el atributo solicitado por el selector correspondiente en el parámetro **Selectors** .

MQZSL_NO_DEVUELTO
El atributo solicitado por el selector correspondiente en el parámetro **Selectors** no se ha devuelto.

La matriz se inicializa con todos los valores como *MQZSL_NOT_RETURNED*. Cuando un componente de servicio de autorización devuelve un atributo, establece el valor adecuado en la matriz en *MQZSL_NOT_RETURNED*. Esto permite que cualquier otro componente de servicio de autorización, al que se realiza la llamada de consulta, identifique qué atributos ya se han devuelto.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
No hay espacio suficiente para los atributos de carácter.

MQRC_INT_COUNT_TOO_SMALL
No hay suficiente espacio para atributos enteros.

Si *CompCode* es MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR
El número de selectores no es válido.

MQRC_SELECTOR_ERROR
Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
Se han especificado demasiados selectores.

MQRC_INT_ATTR_COUNT_ERROR
El número de atributos enteros no es válido.

MQRC_INT_ATTRS_ARRAY_ERROR
Matriz de atributos enteros no válida.

MQRC_CHAR_ATTR_LENGTH_ERROR
El número de atributos de tipo carácter no es válido.

MQRC_CHAR_ATTRS_ERROR
La serie de atributos de tipo carácter no es válida.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                      component */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE-Renovar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_3 y la invoca el gestor de colas para renovar la lista de autorizaciones mantenidas internamente por el componente.

El identificador de función para esta función (para MQZEP) es MQZID_REFRESH_CACHE (8L).

Sintaxis

MQZ_REFRESH_CACHE(*QMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY-Establecer autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

Sintaxis

MQZ_SET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentDatarname>
Tipo: MQBYTExComponentDataLength -entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY_2 -Establecer autorización (ampliado)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

MQZ_SET_AUTHORITY_2 es como MQZ_SET_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData** .

Sintaxis

MQZ_SET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización para el objeto se va a establecer. Para obtener más detalles, consulte "MQZED-Descriptor de entidad" en la página 1724.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto sobre el que se va a establecer la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

Autorización
Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY-Terminar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_AUTHORITY.

Sintaxis

MQZ_TERM_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY
Terminación primaria.

MQZTO_SECUNDARIO
Terminación secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDataen la llamada MQZ_INIT_AUTHORITY.

Cuando se ha completado la llamada MQZ_TERM_AUTHORITY, el gestor de colas descarta estos datos.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_TERMINATION_FAILED
(2287, X'8FF') La terminación ha fallado por una razón no definida.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_NAME-Suprimir nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para suprimir una entrada para la cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_NAME.

Sintaxis

MQZ_DELETE_NAME(*QMgrName* , *QName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a suprimir una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData

Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDataen la llamada MQZ_INIT_NAME.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Tiene que ser uno de los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_STOP
No continúe con el componente siguiente.

Para el mandato **MQZ_DELETE_NAME** , el gestor de colas no intenta iniciar otro componente, independientemente de lo que se devuelva en el parámetro **Continuation** .

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_UNKNOWN_NAME
(2288, X'8F0') No se ha encontrado el nombre de cola.

Nota: Es posible que no sea posible devolver este código si el servicio subyacente responde correctamente en este caso.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_NAME-Inicializar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_NAME.

Sintaxis

MQZ_INIT_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY
Inicialización primaria.

MQZIO_SECUNDARIO
Inicialización secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

Longitud de ComponentData
Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData
Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Versión
Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en el momento de la devolución el gestor de colas no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_NAME del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1
Versión 1.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') La inicialización ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

MQZ_INSERT_NAME-Insertar nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para insertar una entrada para la cola especificada, que contiene el nombre del gestor de colas propietario de la cola. Si la cola ya está definida en el servicio, la llamada falla.

El identificador de función para esta función (para MQZEP) es MQZID_INSERT_NAME.

Sintaxis

MQZ_INSERT_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a insertar una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName
Tipo: MQCHAR48 -entrada

Nombre de gestor de colas resuelto. El nombre del gestor de colas en el que se resuelve la cola. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

continuación
Tipo: MQLONG-entrada/salida

Indicador de continuación establecido por componente. Para MQZ_INSERT_NAME, el gestor de colas no intenta iniciar otro componente, lo que se devuelva en el parámetro **Continuation** .

Se admiten los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS
(2290, X'8F2') El objeto de cola ya existe.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQCHAR48  ResolvedQMgrName;  /* Resolved queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_LOOKUP_NAME-Nombre de búsqueda

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para recuperar el nombre del gestor de colas propietario, para una cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_LOOKUP_NAME.

Sintaxis

MQZ_LOOKUP_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a resolver una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName

Tipo: MQCHAR48 -salida

Nombre de gestor de colas resuelto. Si la función se completa correctamente, este es el nombre del gestor de colas propietario de la cola.

El nombre devuelto por el componente de servicio debe rellenarse a la derecha con espacios en blanco hasta la longitud completa del parámetro; el nombre no debe terminar con un carácter nulo, ni contener espacios en blanco iniciales o intercalados.

ComponentData

Tipo: MQBYTExComponentDataLength -entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Para MQZ_LOOKUP_NAME, el gestor de colas especifica si se debe iniciar otro componente de servicio de nombres, como se indica a continuación:

- Si *CompCode* es MQCC_OK, no se inician más componentes, independientemente del valor que se devuelva en *Continuación*.
- Si *CompCode* no es MQCC_OK, se inicia un componente adicional, a menos que *Continuation* sea MQZCI_STOP.

Se admiten los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_Q_NAME
(2288, X'8F0') No se ha encontrado el nombre de cola.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
```

```
MQCHAR48  QName;             /* Queue name */
MQCHAR48  ResolvedQMgrName;  /* Resolved queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_NAME-Terminar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_NAME.

Sintaxis

MQZ_TERM_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Lo utiliza el componente al llamar al gestor de colas con la función MQZEP.

Opciones

Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY
Terminación primaria.

MQZTO_SECUNDARIO
Terminación secundaria.

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData

Tipo: MQBYTE x ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

Los datos de componente están en memoria compartida accesible para todos los procesos.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

Cuando se ha completado la llamada MQZ_TERM_NAME, el gestor de colas descarta estos datos.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_TERMINATION_FAILED
(2287, X'8FF') La terminación ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
              &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC-Contexto de aplicación

La estructura MQZAC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro *ApplicationContext* . Este parámetro especifica los datos relacionados con la aplicación de llamada.

Tabla 1 resume los campos de la estructura.

Tabla 838. Campos en MQZAC

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Número de versión de la estructura
ProcessId	Identificador proceso
ThreadId	Identificador de hebra
ApplName	Nombre de la aplicación
UserID	Identificador de usuario
ID deEffectiveUser	Identificador de usuario efectivo

Tabla 838. Campos en MQZAC (continuación)	
Campo	**Descripción**
Entorno	Entorno
CallerType	Tipo de interlocutor
AuthenticationType	Tipo de autenticación
BindType	tipo de enlace

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAC_STRUC_ID

Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAC_VERSION_1

Estructura de contexto de aplicación Version-1 . La constante MQZAC_CURRENT_VERSION especifica el número de versión de la versión actual.

ProcessId

Tipo: MQPID-entrada

Identificador de proceso de la aplicación.

ThreadId

Tipo: MQTID-entrada

Identificador de hebra de la aplicación.

ApplName

Tipo: MQCHAR28 -entrada

Nombre de aplicación.

UserID

Tipo: MQCHAR12 -entrada

Identificador de usuario. En UNIX , este campo especifica el ID de usuario real de la aplicación. En Windows , este campo especifica el ID de usuario de la aplicación.

ID de EffectiveUser

Tipo: MQCHAR12 -entrada

Identificador de usuario efectivo. En UNIX , este campo especifica el ID de usuario efectivo de la aplicación. En Windows , este campo está en blanco.

Entorno

Tipo: MQLONG - entrada

Medio ambiente. Este campo especifica el entorno desde el que se ha realizado la llamada. El campo es uno de los valores siguientes:

MQXE_COMMAND_SERVER
Servidor de mandatos

MQXE_MQSC
Intérprete de mandatos **runmqsc**

MQXE_MCA
Agente de canal de mensajes MQXE_OTHER

MQXE_OTHER
Entorno no definido

CallerType
Tipo: MQLONG - entrada

Tipo de interlocutor. Este campo especifica el tipo de programa que ha realizado la llamada. El campo es uno de los valores siguientes:

MQXACT_EXTERNAL
La llamada es externa al gestor de colas.

MQXACT_INTERNAL
La llamada es interna al gestor de colas.

AuthenticationType
Tipo: MQLONG - entrada

Tipo de autenticación. Este campo especifica el tipo de autenticación que se está realizando. El campo es uno de los valores siguientes:

MQZAT_INITIAL_CONTEXT
La llamada de autenticación se debe a que se está inicializando el contexto de usuario. Este valor se utiliza durante una llamada MQCONN o MQCONNX.

CONTEXTO_CAMBIO_MQZAT_CONTEXTO
La llamada de autenticación se debe a que se ha cambiado el contexto de usuario. Este valor se utiliza cuando el MCA cambia el contexto de usuario. Tema principal: MQZAC-

BindType
Tipo: MQLONG - entrada

Tipo de enlace. Este campo especifica el tipo de enlace en uso. El campo es uno de los valores siguientes:

MQCNO_FASTPATH_BINDING
Enlace de vía de acceso rápida.

MQCNO_SHARED_BINDING
Enlace compartido.

MQCNO_ISOLATED_BINDING
Enlace aislado.

Declaración C

Declare los campos de la estructura como se indica a continuación:

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQPID     ProcessId;          /* Process identifier */
  MQTID     ThreadId;           /* Thread identifier */
  MQCHAR28  ApplName;           /* Application name */
  MQCHAR12  UserID;             /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;        /* Environment */
  MQLONG    CallerType;         /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
```

```
  MQLONG    BindType;           /* Bind type */
};
```

MQZAD-Datos de autorización

La estructura MQZAD se utiliza en la llamada MQZ_ENUMERATE_AUTORITY_DATA para dos parámetros, uno de entrada y otro de salida.

Consulte "MQZ_ENUMERATE_AUTORITY_DATA-Enumerar datos de autorización" en la página 1681 para obtener más información sobre los parámetros **Filter** y **AuthorityBuffer** :

- MQZAD se utiliza para el parámetro **Filter** que se especifica en la llamada. Este parámetro especifica los criterios de selección que deben utilizarse para seleccionar los datos de autorización devueltos por la llamada.
- MQZAD también se utiliza para el parámetro **AuthorityBuffer** que es la salida de la llamada. Este parámetro especifica las autorizaciones para una combinación de nombre de perfil, tipo de objeto y entidad.

Tabla 1. resume los campos de la estructura.

Tabla 839. Campos en MQZAD

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Número de versión de la estructura
ProfileName	Nombre de perfil
ObjectType	Tipo de objeto
Autorización	Autorización
EntityDataPtr	Puntero a datos de entidad
EntityType	Tipo de entidad
Opciones	Opciones

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAD_STRUC_ID

Identificador de la estructura de datos de autorización.

Para el lenguaje de programación C, también se define la constante MQZAD_STRUC_ID_ARRAY; tiene el mismo valor que MQZAD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAD_VERSION_1

Estructura de contexto de aplicación Version-1 . La constante MQZAD_CURRENT_VERSION especifica el número de versión de la versión actual.

La constante siguiente especifica el número de versión de la versión actual:

MQZAD_CURRENT_VERSION

Versión actual de la estructura de datos de autorización.

ProfileName
Tipo: MQCHAR48 -entrada

Nombre de perfil.

Para el parámetro **Filter** , este campo es el nombre de perfil para el que se necesitan datos de autorización. Si el nombre está completamente en blanco hasta el final del campo o el primer carácter nulo, se devuelven los datos de autorización para todos los nombres de perfil.

Para el parámetro **AuthorityBuffer** , este campo es el nombre de un perfil que coincide con los criterios de selección especificados.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto.

Para el parámetro **Filter** , este campo es el tipo de objeto para el que se necesitan datos de autorización. Si el valor es MQOT_ALL, se devuelven los datos de autorización para todos los tipos de objeto.

Para el parámetro **AuthorityBuffer** , este campo es el tipo de objeto al que se aplica el perfil identificado por el parámetro **ProfileName** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQOT_ALL también es válido:

MQOT_AUTH_INFO
Información de autenticación

MQOT_CHANNEL
Canal

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente

MQOT_ESCUCHA
Escucha

MQOT_NAMELIST
Lista de nombres

MQOT_PROCESS
Definición de proceso

MQOT_Q
Cola

MQOT_Q_MGR
Gestor de colas

SERVICIO MQOT_SERVICE
Servicio

Autorización
Tipo: MQLONG - entrada

Autorización.

Para el parámetro **Filter** , este campo se ignora.

Para el parámetro **AuthorityBuffer** , este campo representa las autorizaciones que la entidad tiene para los objetos identificados por **ProfileName** y **ObjectType**. Si la entidad sólo tiene una autorización, el campo es igual al valor de autorización adecuado (constante MQZAO_ *). Si la entidad tiene más de una autorización, el campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

EntityDataPtr
Tipo: PMQZED-entrada

Dirección de la estructura MQZED que identifica una entidad.

Para el parámetro **Filter** , este campo apunta a una estructura MQZED que identifica la entidad para la que se necesitan datos de autorización. Si **EntityDataPtr** es el puntero nulo, se devuelven los datos de autorización para todas las entidades.

Para el parámetro **AuthorityBuffer** , este campo apunta a una estructura MQZED que identifica la entidad para la que se han devuelto datos de autorización.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad.

Para el parámetro **Filter** , este campo especifica el tipo de entidad para el que se necesitan datos de autorización. Si el valor es MQZAET_NONE, se devuelven los datos de autorización para todos los tipos de entidad.

Para el parámetro **AuthorityBuffer** , este campo especifica el tipo de la entidad identificada por la estructura MQZED a la que apunta el parámetro **EntityDataPtr** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQZAET_NONE también es válido:

MQZAET_PRINCIPAL
Entidad de seguridad

GRUPO_MQZAC
Grupo

Opciones
Tipo: MQAUTHOPT-entrada

Opciones. Este campo especifica las opciones que dan control sobre los perfiles que se visualizan. Se debe especificar uno de los valores siguientes:

MQAUTHOPT_NAME_ALL_MATCHING
Muestra todos los perfiles

MQAUTHOPT_NAME_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre que el especificado en el campo **ProfileName** .

Además, también debe especificarse uno de los siguientes:

MQAUTHOPT_ENTITY_SET
Visualizar todos los perfiles que se utilizan para calcular la autorización acumulativa que la entidad tiene sobre el objeto especificado por el parámetro **ProfileName** . El parámetro **ProfileName** no debe contener ningún carácter comodín.

- Si la entidad especificada es un principal, para cada miembro del conjunto {entity, groups} se muestra el perfil más aplicable que se aplica al objeto.
- Si la entidad especificada es un grupo, se visualiza el perfil más aplicable del grupo que se aplica al objeto.
- Si se especifica este valor, los valores de **ProfileName**, **ObjectType**, **EntityType**y el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** no deben estar en blanco.

Si ha especificado MQAUTHOPT_NAME_ALL_MATCHING, también puede especificar el valor siguiente:

MQAUTHOPT_ENTITY_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre de entidad que el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** .

Declaración C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

MQZED-Descriptor de entidad

La estructura MQZED se utiliza en un número de llamadas de servicio de autorización para especificar la entidad para la que se va a comprobar la autorización.

Tabla 1. resume los campos de la estructura.

Tabla 840. Campos en MQZED	
Campo	**Descripción**
StrucId	Identificador de la estructura
Versión	Versión
EntityName Ptr	Nombre de entidad
EntityDomainPtr	Puntero de dominio de entidad
SecurityId	Identificador de seguridad
CorrelationPtr	Puntero de correlación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZED_STRUC_ID
Identificador de la estructura del descriptor de entidad.

Para el lenguaje de programación C, también se define la constante MQZED_STRUC_ID_ARRAY; tiene el mismo valor que MQZED_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZED_VERSION_1
Estructura del descriptor de entidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZED_CURRENT_VERSION
Versión actual de la estructura del descriptor de entidad.

EntityNamePtr

Tipo: PMQCHAR-entrada

Nombre de perfil.

Dirección del nombre de entidad. Se trata de un puntero al nombre de la entidad cuya autorización se va a comprobar.

EntityDomainPtr
Tipo: PMQCHAR-entrada

Dirección del nombre de dominio de entidad. Es un puntero al nombre del dominio que contiene la definición de la entidad cuya autorización se va a comprobar.

SecurityId
Tipo: MQBYTE40 -entrada

Autorización.

Identificador de seguridad. Es el identificador de seguridad cuya autorización se debe comprobar.

CorrelationPtr
Tipo: MQPTR-entrada

Puntero de correlación. Esto facilita el paso de datos correlacionales entre la función de autenticar usuario y otras funciones de OAM adecuadas.

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

MQZEP-Añadir punto de entrada de componente

Un componente de servicio inicia esta función, durante la inicialización, para añadir un punto de entrada al vector de punto de entrada para dicho componente de servicio.

Sintaxis

MQZEP (*Hconfig* , *Función* , *EntryPoint* , *CompCode* , *Razón*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente que se está configurando para este servicio instalable en particular. Debe ser el mismo que el componente que el gestor de colas ha pasado a la función de configuración de componente en la llamada de inicialización de componente.

Función
Tipo: MQLONG - entrada

Identificador de función. Los valores válidos para esto se definen para cada servicio instalable.

Si se llama a MQZEP más de una vez para la misma función, la última llamada realizada proporciona el punto de entrada que se utiliza.

EntryPoint
Tipo: PMQFUNC-entrada

Punto de entrada de función. Es la dirección del punto de entrada proporcionado por el componente para realizar la función.

El valor NULL es válido e indica que este componente no proporciona la función. Se asume NULL para puntos de entrada que no están definidos utilizando MQZEP.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_ERROR
(2281, X'8E9') El identificador de función no es válido.

MQRC_HCONFIG_ERROR
(2280, X'8E8') El descriptor de contexto de configuración no es válido.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQZFP-Parámetros libres

La estructura MQZFP se utiliza en la llamada MQZ_FREE_USER para el parámetro *FreeParms* . Este parámetro especifica los datos relacionados con el recurso que se va a liberar.

Tabla 1. resume los campos de la estructura.

Tabla 841. Campos en MQZFP

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Versión
Reserved	Reservado, campo
CorrelationPtr	Puntero de correlación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID

Identificador de la estructura de contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZFP_VERSION_1

Estructura de parámetros libres Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZFP_CURRENT_VERSION

Versión actual de la estructura de parámetros libres.

Reserved

Tipo: MQBYTE8 -entrada

Campo reservado. El valor inicial es nulo.

CorrelationPtr

Tipo: MQPTR-entrada

Puntero de correlación. Dirección de los datos de correlación relacionados con el recurso que se va a liberar.

Declaración C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

MQZIC-Contexto de identidad

La estructura MQZIC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro *IdentityContext* .

La estructura MQZIC contiene información de contexto de identidad, que identifica al usuario de la aplicación que colocó primero el mensaje en una cola:

- El gestor de colas rellena el campo *UserIdentifier* con un nombre que identifica al usuario, la forma en que el gestor de colas puede hacerlo depende del entorno en el que se ejecuta la aplicación.
- El gestor de colas rellena el campo *AccountingToken* con una señal o un número que ha determinado a partir de la aplicación que ha colocado el mensaje.
- Las aplicaciones pueden utilizar el campo *Datos deApplIdentity* para cualquier información adicional que deseen incluir sobre el usuario (por ejemplo, una contraseña cifrada).

Las aplicaciones debidamente autorizadas pueden establecer el contexto de identidad utilizando la función MQZ_AUTHENTICATE_USER.

Un identificador de seguridad de sistemas Windows (SID) se almacena en el campo *AccountingToken* cuando se crea un mensaje en IBM MQ for Windows. El SID se puede utilizar para complementar el campo *UserIdentifier* y para establecer las credenciales de un usuario.

Tabla 1. resume los campos de la estructura.

Tabla 842. Campos en MQZIC	
Campo	**Descripción**
StrucId	Identificador de la estructura
Versión	Versión
UserIdentifier	Identificador de usuario
AccountingToken	Señal de contabilidad
ApplIdentityData	Datos de identidad de la aplicación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID
Identificador de la estructura de contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZIC_VERSION_1
Estructura de contexto de identidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZIC_CURRENT_VERSION
Versión actual de la estructura de contexto de identidad.

UserIdentifier

Tipo: MQCHAR12 -entrada

Identificador de usuario. Esto forma parte del contexto de identidad del mensaje. *UserIdentifier* especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Para obtener más información sobre el campo *UserIdentifier* , consulte “UserIdentifier (MQCHAR12)” en la página 459.

AccountingToken

Tipo: MQBYTE32 -entrada

Señal de contabilidad. Esto forma parte del contexto de identidad del mensaje. *AccountingToken* permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cargue correctamente. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido. Para obtener más información sobre el campo *AccountingToken* , consulte “AccountingToken (MQBYTE32)” en la página 461.

ApplIdentityData

Tipo: MQCHAR32 -entrada

Datos de la aplicación relacionados con la identidad. Esto forma parte del contexto de identidad del mensaje. ApplIdentityLos datos son información definida por la suite de aplicaciones que se

puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza. Para obtener más información sobre el campo de datos ApplIdentity, consulte “Datos de ApplIdentity(MQCHAR32)” en la página 462.

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

IBM i Información de consulta sobre la interfaz de servicios instalables en IBM i

Utilice esta información para comprender la información de referencia para los servicios instalables para IBM i.

Para cada función hay una descripción, incluido el identificador de función (para MQZEP).

Los *parámetros* se muestran en el orden en el que deben aparecer. Todos ellos deben estar presentes.

Cada nombre de parámetro va seguido de su tipo de datos entre paréntesis. Estos son los tipos de datos elementales que se describen en “Tipos de datos elementales” en la página 1017.

También se proporciona la invocación del lenguaje C, después de la descripción de los parámetros.

Conceptos relacionados

IBM i Servicios y componentes instalables para IBM i

ULW Servicios y componentes instalables para UNIX, Linux y Windows

Referencia relacionada

“Información de consulta sobre la interfaz de servicios instalables” en la página 1664
Esta colección de temas proporciona información de referencia para los servicios instalables.

IBM i MQZEP (Añadir punto de entrada de componente) en IBM i

Esta función la invoca un componente de servicio, durante la inicialización, para añadir un punto de entrada al vector de punto de entrada para dicho componente de servicio.

Sintaxis

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

Parámetros

La llamada MQZEP tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada

Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente que se está configurando para este servicio instalable en particular. Debe ser el mismo que el que el gestor de colas ha pasado a la función de configuración de componente en la llamada de inicialización de componente.

Función (MQLONG)-entrada
Identificador de función.

Los valores válidos para esto se definen para cada servicio instalable. Si se llama a MQZEP más de una vez para la misma función, la última llamada realizada proporciona el punto de entrada que se utiliza.

EntryPoint (PMQFUNC)-entrada
Punto de entrada de función.

Es la dirección del punto de entrada proporcionado por el componente para realizar la función. El valor NULL es válido e indica que este componente no proporciona la función. Se asume NULL para puntos de entrada que no están definidos utilizando MQZEP.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_ERROR
(2281, X'8E9') El identificador de función no es válido.

MQRC_HCONFIG_ERROR
(2280, X'8E8') El descriptor de contexto de configuración no es válido.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

IBM i MQHCONFIG (descriptor de contexto de configuración) en IBM i

El tipo de datos MQHCONFIG representa un descriptor de contexto de configuración, es decir, el componente que se está configurando para un servicio instalable determinado. Un descriptor de contexto de configuración debe estar alineado en su límite natural.

Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

Declaración C

```
typedef void MQPOINTER MQHCONFIG;
```

IBM i PMQFUNC (Puntero para función) en IBM i

Puntero a una función.

Declaración C

```
typedef void MQPOINTER PMQFUNC;
```

IBM i MQZ_AUTHENTICATE_USER (Autenticar usuario) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 . Lo invoca el gestor de colas para autenticar un usuario o para establecer campos de contexto de identidad.

Se invoca cuando se establece un contexto de aplicación de usuario IBM MQ . Esto sucede durante las llamadas de conexión en el punto en el que se inicializa el contexto de usuario de la aplicación y en cada punto en el que se cambia el contexto de usuario de la aplicación. Cada vez que se realiza una llamada de conexión, la información de contexto de usuario de la aplicación se vuelve a adquirir en el campo *IdentityContext* .

El identificador de función para esta función (para MQZEP) es MQZID_AUTHENTICATE_USER.

Sintaxis

MQZ_AUTHENTICATE_USER ***(QMgrName, SecurityParms, ApplicationContext,*** *IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason)*

Parámetros

La llamada MQZ_AUTHENTICATE_USER tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo. El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SecurityParms (MQCSP)-entrada
Parámetros de seguridad.

Datos relacionados con el ID de usuario, la contraseña y el tipo de autenticación.

Durante una llamada MQCONN MQI, este parámetro contiene valores nulos o predeterminados.

ApplicationContext (MQZAC)-entrada
Contexto de aplicación.

Datos relacionados con la aplicación de llamada. Consulte “MQZAC (contexto de aplicación) en IBM i” en la página 1761 para obtener los detalles. Durante cada llamada MQCONN o MQCONNX MQI, se vuelve a adquirir la información de contexto de usuario de la estructura MQZAC.

IdentityContext (MQZIC)-entrada/salida
Contexto de identidad.

En la entrada de la función de autenticar usuario, identifica el contexto de identidad actual. La función de autenticar usuario puede cambiar esto, momento en el que el gestor de colas adopta el nuevo

contexto de identidad. Consulte “MQZIC (contexto de identidad) en IBM i” en la página 1768 para obtener más detalles sobre la estructura MQZIC.

CorrelationPtr (MQPTR)-salida
Puntero de correlación.

Especifica la dirección de cualquier dato de correlación. A continuación, este puntero se pasa a otras llamadas de OAM.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios que realice en él cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componentes. La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Distintivo de continuación.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
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 (Comprobar autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

Sintaxis

MQZ_CHECK_AUTHORITY ***(QMgrName, EntityName, EntityType,***
ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_CHECK_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo. El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada
Nombre de entidad.

El nombre de la entidad cuya autorización para el objeto se va a comprobar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType (MQLONG)-entrada
Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Es uno de los siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización (MQLONG)-entrada
Autorización que se debe comprobar.

Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

Las autorizaciones siguientes se aplican al uso de las llamadas MQI:

MQZAO_CONNECT
Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE
Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT
Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT
Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE
Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET
Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT
Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ IDENTITY_ CONTEXT
Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ ALL_CONTEXT
Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI
Todas las autorizaciones MQI.

Esto habilita todas las autorizaciones descritas anteriormente.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR
Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE
Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY
Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE
Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR
Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR
Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL
Capacidad para iniciar, detener o hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED
Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN
Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las autorizaciones siguientes se aplican tanto al uso de la MQI como a la administración de un gestor de colas:

MQZAO_TODOS
Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NONE
Sin autorizaciones.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
```

```
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-Comprobar si el usuario tiene privilegios

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_6 y la invoca el gestor de colas para determinar si un usuario especificado es un usuario privilegiado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_PRIVILEGED.

Sintaxis

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad que se va a comprobar. Para obtener más información, consulte “MQZED-Descriptor de entidad” en la página 1724.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityData. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ComponentData

Tipo: MQBYTExComponentDataLength -entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación* .

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NO_PRIVILEGIADO
(2584, X'A18') Este usuario no es un ID de usuario privilegiado.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de terminación y de razón de la API.

Invocación en C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_COPY_ALL_AUTHORITY (Copiar todas las autorizaciones) en IBM i

Esta función la proporciona un componente de servicio de autorización. Lo invoca el gestor de colas para copiar todas las autorizaciones que están actualmente en vigor para un objeto de referencia en otro objeto.

El identificador de función para esta función (para MQZEP) es MQZID_COPY_ALL_AUTHORITY.

Sintaxis

MQZ_COPY_ALL_AUTHORITY *(QMgrName, RefObjectName, ObjectName, ObjectType, ComponentData, Continuation, CompCode, Reason)*

Parámetros

La llamada MQZ_COPY_ALL_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

RefObjectName (MQCHAR48)-entrada
Nombre de objeto de referencia.

El nombre del objeto de referencia, cuyas autorizaciones se van a copiar. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto para el que se van a establecer los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de objeto especificado por *RefObjectName* y *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_COPY_ALL_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Objeto de referencia desconocido.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_DELETE_AUTHORITY (Suprimir autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas para suprimir todas las autorizaciones asociadas con el objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_AUTHORITY.

Sintaxis

MQZ_DELETE_AUTHORITY ***(QMgrName, ObjectName, ObjectType,***
ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_DELETE_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto para el que se van a suprimir los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_DELETE_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_ENUMERATE_AUTORITY_DATA (Enumerar datos de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_4 y la invoca repetidamente el gestor de colas para recuperar todos los datos de autorización que coinciden con los criterios de selección especificados en la primera invocación.

El identificador de función para esta función (para MQZEP) es MQZID_ENUMERATE_AUTORITY_DATA.

Sintaxis

MQZ_ENUMERATE_AUTHORITY_DATA *(**QMgrName, StartEnumeration,** Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength, ComponentData, Continuation, CompCode, Reason)*

Parámetros

La llamada MQZ_ENUMERATE_AUTORITY_DATA tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

StartEnumeration (MQLONG)-entrada
Distintivo que indica si la llamada debe iniciar la enumeración.

Esto indica si la llamada debe iniciar la enumeración de datos de autorización o continuar la enumeración de datos de autorización iniciada por una llamada anterior a MQZ_ENUMERATE_AUTORITY_DATA. El valor puede ser uno de los siguientes:

MQZSE_START
Iniciar enumeración.

La llamada se invoca con este valor para iniciar la enumeración de datos de autorización. El parámetro **Filter** especifica los criterios de selección que se deben utilizar para seleccionar los datos de autorización devueltos por esta y por llamadas sucesivas.

MQZSE_CONTINUE
Continuar enumeración.

La llamada se invoca con este valor para continuar la enumeración de datos de autorización. El parámetro **Filter** se ignora en este caso y se puede especificar como puntero nulo (los criterios

de selección se determinan mediante el parámetro **Filter** especificado por la llamada que tenía *StartEnumeration* establecido en MQZSE_START).

Filtro (MQZAD)-entrada
Filtro.

Si *StartEnumeration* es MQZSE_START, *Filter* especifica los criterios de selección que se van a utilizar para seleccionar los datos de autorización que se van a devolver. Si *Filter* es el puntero nulo, no se utiliza ningún criterio de selección, es decir, se devuelven todos los datos de autorización. Consulte "MQZAD (datos de autorización) en IBM i" en la página 1763 para obtener detalles de los criterios de selección que se pueden utilizar.

Si *StartEnumeration* es MQZSE_CONTINUE, *Filter* se ignora y se puede especificar como puntero nulo.

AuthorityBufferLength (MQLONG)-entrada
Longitud de *AuthorityBuffer*.

Es la longitud en bytes del parámetro **AuthorityBuffer** . El almacenamiento intermedio de autorización debe ser lo suficientemente grande para acomodar los datos que se van a devolver.

AuthorityBuffer (MQZAD)-salida
Datos de autorización.

Es el almacenamiento intermedio en el que se devuelven los datos de autorización. El almacenamiento intermedio debe ser lo suficientemente grande para acomodar una estructura MQZAD, una estructura MQZED, más el nombre de entidad más largo y el nombre de dominio más largo definido.

Nota: Este parámetro se define como MQZAD, ya que MQZAD siempre se produce al principio del almacenamiento intermedio. Sin embargo, si el almacenamiento intermedio se declara realmente como MQZAD, el almacenamiento intermedio será demasiado pequeño; debe ser más grande que un MQZAD para que pueda acomodar los nombres de entidad y dominio MQZAD, MQZED.

AuthorityDataLength (MQLONG)-salida
Longitud de los datos devueltos en *AuthorityBuffer*.

Es la longitud de los datos devueltos en *AuthorityBuffer*. Si el almacenamiento intermedio de autorización es demasiado pequeño, *AuthorityDataLength* se establece en la longitud del almacenamiento intermedio necesaria y la llamada devuelve el código de terminación MQCC_FAILED y el código de razón MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_ENUMERATE_AUTORITY_DATA, esto tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') No hay datos disponibles.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;               /* Queue manager name */
MQLONG    StartEnumeration;       /* Flag indicating whether call should
                                     start enumeration */
MQZAD     Filter;                 /* Filter */
MQLONG    AuthorityBufferLength;  /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;        /* Authority data */
MQLONG    AuthorityDataLength;    /* Length of data returned in
                                     AuthorityBuffer */
MQBYTE    ComponentData[n];       /* Component data */
MQLONG    Continuation;           /* Continuation indicator set by
                                     component */
MQLONG    CompCode;               /* Completion code */
MQLONG    Reason;                 /* Reason code qualifying CompCode */
```

MQZ_FREE_USER-Usuario libre

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para liberar el recurso asignado asociado. Se invoca cuando una aplicación ha terminado de ejecutarse en todos los contextos de usuario, por ejemplo durante una llamada MQI MQDISC.

El identificador de función para esta función (para MQZEP) es MQZID_FREE_USER.

IBM i MQZ_GET_AUTHORITY (Obtener autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

Sintaxis

MQZ_GET_AUTHORITY ***(QMgrName, EntityName, EntityType, ObjectName,***
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_GET_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada
Nombre de entidad.

El nombre de la entidad cuyo acceso al objeto se va a recuperar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada
Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto para el que se va a recuperar la autorización de la entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización (MQLONG)-salida
Autoridad de entidad.

Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_GET_EXPLICIT_AUTHORITY (Obtener autorización explícita) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que el grupo primario del principal con nombre tiene para acceder a un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY ***(QMgrName, EntityName, EntityType,*** *ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Parámetros

La llamada MQZ_GET_EXPLICIT_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada
Nombre de entidad.

El nombre de la entidad desde la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada
Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto para el que se va a recuperar la autorización de la entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización (MQLONG)-salida
Autoridad de entidad.

Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_EXPLICIT_AUTHORITY esto tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
```

```
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_INIT_AUTHORITY (inicializar servicio de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_AUTHORITY.

Sintaxis

MQZ_INIT_AUTHORITY ***(Hconfig, Options, QMgrName, ComponentDataLength,***
ComponentData, Version, CompCode, Reason)

Parámetros

La llamada MQZ_INIT_AUTHORITY tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada
Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones (MQLONG)-entrada
Opciones de inicialización.

Es uno de los siguientes:

MQZIO_PRIMARY
Inicialización primaria.

MQZIO_SECUNDARIO
Inicialización secundaria.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLength (MQLONG)-entrada
Longitud de los datos de componente.

Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

Versión (MQLONG)-entrada/salida
Número de versión.

En la entrada de la función de inicialización, identifica el número de versión *más alto* al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que *da soporte* . Si en el retorno el gestor de colas no da soporte a la versión devuelta por el componente, llama a la función MQZ_TERM_AUTHORITY del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1
Versión 1.

MQZAS_VERSION_2
Versión 2.

MQZAS_VERSION_3
Versión 3.

MQZAS_VERSION_4
Versión 4.

MQZAS_VERSION_5
Versión 5.

MQZAS_VERSION_6
Versión 6.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') La inicialización ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
```

```
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

IBM i MQZ_INQUIRE (Consultar servicio de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para consultar la funcionalidad soportada. Cuando se utilizan varios componentes de servicio, se llama a los componentes de servicio en orden inverso al orden en el que se instalaron.

El identificador de función para esta función (para MQZEP) es MQZID_INQUIRE.

Sintaxis

MQZ_INQUIRE
(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_INQUIRE tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SelectorCount (MQLONG)-entrada
Número de selectores.

Número de selectores proporcionados en el parámetro Selectores.

El valor debe estar entre cero y 256.

Selectores (MQLONG x SelectorCount)-entrada
Selectores.

Matriz de selectores. Cada selector identifica un atributo necesario y debe ser de uno de los tipos siguientes:

- MQIACF_ * (entero)
- MQCACF_ * (carácter)

Los selectores se pueden especificar en cualquier orden. El número de selectores de la matriz se indica mediante el parámetro SelectorCount .

Los atributos enteros identificados por los selectores se devuelven en el parámetro IntAttrs en el mismo orden en que aparecen en los selectores.

Los atributos de carácter identificados por los selectores se devuelven en el parámetro CharAttrs en el mismo orden en que aparecen los selectores.

IntAttrCount (MQLONG)-entrada
Número de atributos enteros.

El número de atributos enteros proporcionados en el parámetro IntAttrs .

El valor debe estar en el rango de 0 a 256.

IntAttrs (MQLONG x IntAttrCount)-salida
Atributos enteros.

Matriz de atributos enteros. Los atributos de entero se devuelven en el mismo orden que los selectores de entero correspondientes en la matriz de selectores.

Recuento de CharAttr(MQLONG)-entrada
Longitud del almacenamiento intermedio de atributos de caracteres.

Longitud en bytes del parámetro CharAttrs .

El valor debe al menos sumar las longitudes de los atributos de caracteres solicitados. Si no se solicita ningún atributo de carácter, cero es un valor válido.

CharAttrs (MQLONG x CharAttrCount)-salida
Almacenamiento intermedio de atributos de caracteres.

Almacenamiento intermedio que contiene atributos de caracteres, concatenados entre sí. Los atributos de carácter se devuelven en el mismo orden que los selectores de caracteres correspondientes en la matriz Selectores.

La longitud del almacenamiento intermedio la proporciona el parámetro de recuento CharAttr.

SelectorReturned (Recuento deMQLONGxSelector)-entrada
Se ha devuelto el selector.

Matriz de valores que identifican qué atributos se han devuelto del conjunto solicitado por los selectores en el parámetro Selectores. El número de valores de esta matriz se indica mediante el parámetro SelectorCount . Cada valor en la matriz se relaciona con el selector desde la posición correspondiente en la matriz de selectores. Cada valor es uno de los siguientes:

MQZSL_DEVUELTO
Se ha devuelto el atributo solicitado por el selector correspondiente en el parámetro Selectores.

MQZSL_NO_DEVUELTO
El atributo solicitado por el selector correspondiente en el parámetro Selectores no se ha devuelto.

La matriz se inicializa con todos los valores como *MQZSL_NOT_RETURNED*. Cuando un componente de servicio de autorización devuelve un atributo, establece el valor adecuado en la matriz en *MQZSL_RETURNED*. Esto permite que cualquier otro componente de servicio de autorización, al que se realiza la llamada de consulta, identifique qué atributos ya se han devuelto.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **`ComponentDataLength`** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Distintivo de continuación.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
No hay espacio suficiente para los atributos de carácter.

MQRC_INT_COUNT_TOO_SMALL
No hay suficiente espacio para atributos enteros.

Si *CompCode* es MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR
El número de selectores no es válido.

MQRC_SELECTOR_ERROR
Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
Se han especificado demasiados selectores.

MQRC_INT_ATTR_COUNT_ERROR
El número de atributos enteros no es válido.

MQRC_INT_ATTRS_ARRAY_ERROR
Matriz de atributos enteros no válida.

MQRC_CHAR_ATTR_LENGTH_ERROR
El número de atributos de tipo carácter no es válido.

MQRC_CHAR_ATTRS_ERROR
La serie de atributos de tipo carácter no es válida.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                   &IntAttrs, CharAttrLength, &CharAttrs,
                   SelectorReturned, ComponentData, &Continuation,
                   &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                      component */
```

```
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

IBM i MQZ_REFRESH_CACHE (Renovar todas las autorizaciones) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_3 . Lo invoca el gestor de colas para renovar la lista de autorizaciones mantenidas internamente por el componente.

El identificador de función para esta función (para MQZEP) es MQZID_REFRESH_CACHE (8L).

Sintaxis

MQZ_REFRESH_CACHE
(QMgrName, ComponentData, Continuación, CompCode, Razón)

Parámetros

***QMgrName* (MQCHAR48)-entrada**
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

***ComponentData* (MQBYTE x *ComponentDataLongitud*) -entrada/salida**
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular. Los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una función del componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

***Continuación* (MQLONG)-salida**
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_REFRESH_CACHE, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

***CompCode* (MQLONG)-salida**
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

***Razón* (MQLONG)-salida**
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_SET_AUTHORITY (Establecer autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

Sintaxis

MQZ_SET_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName,*
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_SET_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada
Nombre de entidad.

El nombre de la entidad para la que se va a establecer el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada
Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName (MQCHAR48)-entrada
El nombre del objeto.

El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada
Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización (MQLONG)-entrada
Autorización que se debe comprobar.

Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se establece más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida
Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_SET_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Invocación en C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY-Terminar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_AUTHORITY.

Sintaxis

MQZ_TERM_AUTHORITY ***(Hconfig, Options, QMgrName, ComponentData,***
CompCode, Reason)

Parámetros

La llamada MQZ_TERM_AUTHORITY tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada
Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente concreto que se está terminando.

Opciones (MQLONG)-entrada
Opciones de terminación.

Es uno de los siguientes:

MQZTO_PRIMARY
Terminación primaria.

MQZTO_SECUNDARIO
Terminación secundaria.

QMgrName (MQCHAR48)-entrada
Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData (MQBYTE x ComponentDataLength)-entrada/salida
Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Cuando se ha completado la llamada MQZ_TERM_AUTHORITY, el gestor de colas descarta estos datos.

CompCode (MQLONG)-salida
Código de terminación.

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_TERMINATION_FAILED
(2287, X'8FF') La terminación ha fallado por una razón no definida.

Para obtener más información sobre estos códigos de razón, consulte Mensajes y códigos de razón.

Invocación en C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZAC (contexto de aplicación) en IBM i

Este parámetro especifica los datos relacionados con la aplicación de llamada.

La estructura MQZAC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro **ApplicationContext** .

Campos

StrucId (MQCHAR4)
Identificador de estructura.

El valor es:

MQZAC_STRUC_ID
Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG
Número de versión de la estructura.

El valor es:

MQZAC_VERSION_1
Estructura de contexto de aplicación Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZAC_CURRENT_VERSION
Versión actual de la estructura de contexto de aplicación.

Es un campo de entrada para el servicio.

ProcessId (MQPID)
Identificador de proceso.

Identificador de proceso de la aplicación.

ThreadId (MQTID)
Identificador de hebra.

Identificador de hebra de la aplicación.

ApplName (MQCHAR28)
Nombre de aplicación.

Nombre de la aplicación.

UserID (MQCHAR12)
Identificador de usuario.

Para sistemas IBM i , el perfil de usuario bajo el que se ha creado el trabajo de aplicación. (En IBM i, cuando se realiza un intercambio de perfil con la API QWTSETP en el trabajo de aplicación, se devuelve el perfil de usuario actual).

ID de EffectiveUser(MQCHAR12)
Identificador de usuario efectivo.

Para sistemas IBM i , el perfil de usuario actual del trabajo de aplicación.

Entorno (MQLONG)
Medio ambiente.

Este campo especifica el entorno desde el que se ha realizado la llamada.

Puede tener uno de los valores siguientes:

MQXE_COMMAND_SERVER
Servidor de mandatos.

MQXE_MQSC
Intérprete de mandatos `runmqsc` .

MQXE_MCA
Agente de canal de mensajes

MQXE_OTHER
Entorno no definido

CallerType (MQLONG)
Tipo de interlocutor.

Este campo especifica el tipo de programa que ha realizado la llamada.

Puede tener uno de los valores siguientes:

MQXACT_EXTERNAL
La llamada es externa al gestor de colas.

MQXACT_INTERNAL
La llamada es interna al gestor de colas.

AuthenticationType (MQLONG)
Tipo de autenticación.

Este campo especifica el tipo de autenticación que se está realizando.

Puede tener uno de los valores siguientes:

MQZAT_INITIAL_CONTEXT
La llamada de autenticación se debe a que se está inicializando el contexto de usuario. Este valor se utiliza durante una llamada `MQCONN` o `MQCONNX` .

CONTEXTO_CAMBIO_MQZAT_CONTEXTO
La llamada de autenticación se debe a que se ha cambiado el contexto de usuario. Este valor se utiliza cuando el MCA cambia el contexto de usuario.

v

BindType (MQLONG
Tipo de enlace.

Este campo especifica el tipo de enlace en uso.

Puede tener uno de los valores siguientes:

MQCNO_FASTPATH_BINDING
Enlace de vía de acceso rápida.

MQCNO_SHARED_BINDING
Enlace compartido.

MQCNO_ISOLATED_BINDING
Enlace aislado.

Declaración C

```
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 (datos de autorización) en IBM i

La estructura MQZAD se utiliza en la llamada MQZ_ENUMERATE_AUTORITY_DATA para dos parámetros.

Consulte “MQZ_ENUMERATE_AUTORITY_DATA (Enumerar datos de autorización) en IBM i” en la página 1743 para obtener más información sobre los parámetros **Filter** y **AuthorityBuffer** :

- MQZAD se utiliza para el parámetro **Filter** que se especifica en la llamada. Este parámetro especifica los criterios de selección que deben utilizarse para seleccionar los datos de autorización devueltos por la llamada.
- MQZAD también se utiliza para el parámetro **AuthorityBuffer** que es la salida de la llamada. Este parámetro especifica las autorizaciones para una combinación de nombre de perfil, tipo de objeto y entidad.

Campos

StrucId (MQCHAR4)
Identificador de estructura.

El valor es:

MQZAD_STRUC_ID
Identificador de la estructura de datos de autorización.

Para el lenguaje de programación C, también se define la constante MQZAD_STRUC_ID_ARRAY; tiene el mismo valor que MQZAD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)
Número de versión de la estructura.

El valor es:

MQZAD_VERSION_1
Estructura de datos de autorización Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZAD_CURRENT_VERSION
Versión actual de la estructura de datos de autorización.

Es un campo de entrada para el servicio.

ProfileName (MQCHAR48)
Nombre de perfil.

Para el parámetro **Filter** , este campo es el nombre de perfil desde el que se necesitan los datos de autorización. Si el nombre está completamente en blanco hasta el final del campo o el primer carácter nulo, se devuelven los datos de autorización para todos los nombres de perfil.

Para el parámetro **AuthorityBuffer** , este campo es el nombre de un perfil que coincide con los criterios de selección especificados.

ObjectType (MQLONG)
Tipo de objeto.

Para el parámetro **Filter** , este campo es el tipo de objeto para el que se necesitan datos de autorización. Si el valor es MQOT_ALL, se devuelven los datos de autorización para todos los tipos de objeto.

Para el parámetro **AuthorityBuffer** , este campo es el tipo de objeto al que se aplica el perfil identificado por **ProfileName** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQOT_ALL también es válido:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

Autorización (MQLONG)
Autorización.

Para el parámetro **Filter** , este campo se ignora.

Para el parámetro **AuthorityBuffer** , este campo representa las autorizaciones que la entidad tiene para los objetos identificados por **ProfileName** y **ObjectType**. Si la entidad sólo tiene una autorización, el campo es igual al valor de autorización adecuado (constante MQZAO_ *). Si la entidad tiene más de una autorización, el campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

EntityDataPtr (PMQZED)
Dirección de la estructura MQZED que identifica una entidad.

Para el parámetro **Filter** , este campo apunta a una estructura MQZED que identifica la entidad de la que se necesitan datos de autorización. Si **EntityDataPtr** es el puntero nulo, se devuelven los datos de autorización para todas las entidades.

Para el parámetro **AuthorityBuffer** , este campo apunta a una estructura MQZED que identifica la entidad de la que proceden los datos de autorización devueltos.

EntityType (MQLONG)
Tipo de entidad.

Para el parámetro **Filter** , este campo especifica el tipo de entidad para el que se necesitan datos de autorización. Si el valor es MQZAET_NONE, se devuelven los datos de autorización para todos los tipos de entidad.

Para el parámetro **AuthorityBuffer** , este campo especifica el tipo de la entidad identificada por la estructura MQZED a la que apunta **EntityDataPtr**.

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQZAET_NONE también es válido:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

Opciones (MQAUTHOPT)
Opciones.

Este campo especifica las opciones que dan control sobre los perfiles que se visualizan.

Se debe especificar uno de los siguientes:

MQAUTHOPT_NAME_ALL_MATCHING
Muestra todos los perfiles

MQAUTHOPT_NAME_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre que el especificado en el campo **ProfileName** .

Además, también debe especificarse uno de los siguientes:

MQAUTHOPT_ENTITY_SET
Visualizar todos los perfiles utilizados para calcular la autorización acumulativa que la entidad tiene sobre el objeto especificado por **ProfileName**. El campo **ProfileName** no debe contener ningún carácter comodín.

- Si la entidad especificada es un principal, para cada miembro del conjunto {entity, groups} se muestra el perfil más aplicable que se aplica al objeto.
- Si la entidad especificada es un grupo, se visualiza el perfil más aplicable del grupo que se aplica al objeto.
- Si se especifica este valor, los valores de **ProfileName**, **ObjectType**, **EntityType**y el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** no deben estar en blanco.

Si ha especificado *MQAUTHOPT_NAME_ALL_MATCHING*, también puede especificar lo siguiente:

MQAUTHOPT_ENTITY_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre de entidad que el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** .

Declaración C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
```

```
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

IBM i MQZED (descriptor de entidad) en IBM i

La estructura MQZED se utiliza en un número de llamadas de servicio de autorización para especificar la entidad para la que se va a comprobar la autorización.

Campos

StrucId (MQCHAR4)
Identificador de estructura.

El valor es:

MQZED_STRUC_ID
Identificador de la estructura del descriptor de entidad.

Para el lenguaje de programación C, también se define la constante MQZED_STRUC_ID_ARRAY; tiene el mismo valor que MQZED_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)
Número de versión de la estructura.

El valor es:

MQZED_VERSION_1
Estructura del descriptor de entidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZED_CURRENT_VERSION
Versión actual de la estructura del descriptor de entidad.

Es un campo de entrada para el servicio.

EntityNamePtr (PMQCHAR)
Dirección del nombre de entidad.

Se trata de un puntero al nombre de la entidad cuya autorización se va a comprobar.

EntityDomainPtr (PMQCHAR)
Dirección del nombre de dominio de entidad.

Es un puntero al nombre del dominio que contiene la definición de la entidad cuya autorización se va a comprobar.

SecurityId (MQBYTE40)
Identificador de seguridad.

Es el identificador de seguridad cuya autorización se debe comprobar.

CorrelationPtr (MQPTR)
Puntero de correlación.

Esto facilita el paso de datos correlacionales entre la función de autenticar usuario y otras funciones de OAM adecuadas.

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

IBM i MQZFP (parámetros libres) en IBM i

Este parámetro especifica los datos relacionados con el recurso que se va a liberar.

La estructura MQZFP se utiliza en la llamada MQZ_FREE_USER para el parámetro **FreeParms** .

Campos

StrucId (MQCHAR4)
Identificador de estructura.

El valor es:

MQZFP_STRUC_ID
Identificador de la estructura de parámetros libres.

Para el lenguaje de programación C, la constante MQZFP_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQZFP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)
Número de versión de la estructura.

El valor es:

MQZFP_VERSION_1
Estructura de parámetros libres Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZFP_CURRENT_VERSION
Versión actual de la estructura de parámetros libres.

Es un campo de entrada para el servicio.

Reservado (MQBYTE8)
Campo reservado.

El valor inicial es nulo.

CorrelationPtr (MQPTR)
Puntero de correlación.

Dirección de los datos de correlación relacionados con el recurso que se va a liberar.

Declaración C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

IBM i MQZIC (contexto de identidad) en IBM i

La estructura MQZIC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro **IdentityContext** .

La estructura MQZIC contiene información de contexto de identidad, que identifica al usuario de la aplicación que colocó primero el mensaje en una cola:

- El gestor de colas rellena el campo UserIdentifier con un nombre que identifica al usuario, la forma en que el gestor de colas puede hacerlo depende del entorno en el que se ejecuta la aplicación.
- El gestor de colas rellena el campo AccountingToken con una señal o un número que ha determinado a partir de la aplicación que ha colocado el mensaje.
- Las aplicaciones pueden utilizar el campo de datos ApplIdentitypara cualquier información adicional que deseen incluir sobre el usuario (por ejemplo, una contraseña cifrada).

Las aplicaciones debidamente autorizadas pueden establecer el contexto de identidad utilizando la función MQZ_AUTHENTICATE_USER.

Un identificador de seguridad (SID) de sistemas Windows se almacena en el campo AccountingToken cuando se crea un mensaje en IBM MQ for Windows. El SID se puede utilizar para complementar el campo UserIdentifier y para establecer las credenciales de un usuario.

Campos

StrucId (MQCHAR4)
Identificador de estructura.

El valor es:

MQZIC_STRUC_ID
Identificador de la estructura de contexto de identidad.

Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)
Número de versión de la estructura.

El valor es:

MQZIC_VERSION_1
Estructura de contexto de identidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZIC_CURRENT_VERSION
Versión actual de la estructura de contexto de identidad.

Es un campo de entrada para el servicio.

UserIdentifier (MQCHAR12)
Identificador de usuario.

Forma parte del **contexto de identidad** del mensaje.

UserIdentifier especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Para obtener más información sobre el campo *UserIdentifier* , consulte "UserIdentifier (MQCHAR12)" en la página 459.

AccountingToken (MQBYTE32)
Señal de contabilidad.

Forma parte del **contexto de identidad** del mensaje.

AccountingToken permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cargue correctamente. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido. Para obtener más información sobre el campo *AccountingToken* , consulte “AccountingToken (MQBYTE32)” en la página 461.

Datos de ApplIdentity(MQCHAR32)
Datos de la aplicación relacionados con la identidad.

Forma parte del **contexto de identidad** del mensaje.

ApplIdentityData es información definida por la suite de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza. Para obtener más información sobre el campo *ApplIdentityData* , consulte “Datos de ApplIdentity(MQCHAR32)” en la página 462.

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

Las clases e interfaces de IBM MQ .NET

Las clases e interfaces de IBM MQ .NET se listan alfabéticamente. Se describen las propiedades, métodos y constructores.

Clase MQAsyncStatus.NET

Utilice MQAsyncStatus para consultar sobre el estado de la actividad MQI anterior; por ejemplo, consultar sobre el éxito de las operaciones de colocación asíncrona anteriores. MQAsyncStatus encapsula las características de la estructura de datos de MQSTS .

Clase

```
System.Object
    |
    └ IBM.WMQ.MQBase
        |
        └ IBM.WMQ.MQBaseObject
            |
            └ IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```


- “Propiedades” en la página 1769
- “Constructores” en la página 1770

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public static int CompCode {get;}

El código de terminación del primer error o aviso.

public static int Reason {get;}

El código de razón del primer error o aviso.

public static int PutSuccessCount {get;}

Número de llamadas put de MQI asíncronas satisfactorias.

public static int PutWarningCount {get;}

Número de llamadas put de MQI asíncronas que se han realizado correctamente con un aviso.

public static int PutFailureCount {get;}

Número de llamadas put de MQI asíncronas fallidas.

public static int ObjectType {get;}

El tipo de objeto para el primer error. Son posibles los siguientes valores:

- `MQC.MQOT_ALIAS_Q`
- `MQC.MQOT_LOCAL_Q`
- `MQC.MQOT_MODEL_Q`
- `MQC.MQOT_Q`
- `MQC.MQOT_REMOTE_Q`
- `MQC.MQOT_TOPIC`
- `0`, lo que significa que no se devuelve ningún objeto

public static string ObjectName {get;}

El nombre de objeto.

public static string ObjectQMgrName {get;}

El nombre del gestor de colas de objetos.

public static string ResolvedObjectName {get;}

El nombre de objeto resuelto.

public static string ResolvedObjectQMgrName {get;}

El nombre del gestor de colas de objeto resuelto.

Constructores

public MQAsyncStatus() throws MQException;

Método constructor, construye un objeto con campos inicializados en cero o en blanco según corresponda.

Clase MQAuthenticationInformationRecord.NET

Utilice `MQAuthenticationInformationRecord` para especificar información sobre un autenticador que se va a utilizar en una conexión de cliente TLS de IBM MQ . `MQAuthenticationInformationRecord` encapsula un registro de información de autenticación, `MQAIR`.

Clase

```
System.Object
    │
    └─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- “Propiedades” en la página 1771
- “Constructores” en la página 1771

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public long Version {get; set;}`

Número de versión de la estructura.

`public long AuthInfoType {get; set;}`

El tipo de información de autenticación. Este atributo debe establecerse en uno de los valores siguientes:

- `OCSP` -La comprobación de estado de revocación de certificados se realiza utilizando OCSP.
- `CRLLDAP` -La comprobación de estado de revocación de certificados se realiza utilizando listas de revocación de certificados en servidores LDAP.

`public string AuthInfoConnName {get; set;}`

El nombre DNS o la dirección IP del host en el que se ejecuta el servidor LDAP, con un número de puerto opcional. Esta palabra clave es necesaria.

`public string LDAPPassword {get; set;}`

La contraseña asociada con el nombre distinguido del usuario que está accediendo al servidor LDAP. Esta propiedad sólo se aplica cuando **`AuthInfoType`** se establece en `CRLLDAP`.

`public string LDAPUserName {get; set;}`

Nombre distinguido del usuario que está accediendo al servidor LDAP. Cuando se establece esta propiedad, `LDAPUserNameLength` y `LDAPUserNamePtr` se establecen automáticamente correctamente. Esta propiedad sólo se aplica cuando `AuthInfoTipo` está establecido en `CRLLDAP`.

`public string OCSPResponderURL {get; set;}`

El URL en el que se puede establecer contacto con el programa de respuesta OCSP. Esta propiedad sólo se aplica cuando `AuthInfoTipo` está establecido en `OCSP`

Este campo distingue entre mayúsculas y minúsculas. Debe empezar con la serie http:// en minúsculas. El resto del URL puede ser sensible a las mayúsculas y minúsculas, en función de la implementación del servidor OCSP.

Constructores

`MQAuthenticationInformationRecord();`

Clase `MQDestination.NET`

Utilice `MQDestination` para acceder a métodos comunes a `MQQueue` y `MQTopic`. `MQDestination` es una clase base abstracta y no se puede instanciar.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQDestination
                                        │
                                        └─ IBM.WMQ.MQQueue
                                        │
                                        └─ IBM.WMQ.MQTopic
```

`public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;`

- “Propiedades” en la página 1772
- “Métodos” en la página 1772
- “Constructores” en la página 1774

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public DateTime CreationDateTime {get;}`

Fecha y hora en que se ha creado la cola o el tema. Originalmente contenida en `MQQueue`, esta propiedad se ha movido a la clase `MQDestination` base.

No hay ningún valor predeterminado.

`public int DestinationType {get;}`

Valor entero que describe el tipo de destino que se está utilizando. Inicializado desde el constructor de subclases, `MQQueue` o `MQTopic`, este valor puede tomar uno de estos valores:

- `MQOT_Q`
- `MQOT_TOPIC`

No hay ningún valor predeterminado.

Métodos

`public void Get(MQMessage message);`
`public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);`
`public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);`

Genera `MQException`.

Obtiene un mensaje de una cola si el destino es un objeto `MQQueue` o de un tema si el destino es un objeto `MQTopic` , utilizando una instancia predeterminada de `MQGetMessageOptions` para realizar la obtención.

Si la obtención falla, el objeto `MQMessage` no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del `MQMessage` se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un `MQQueueManager` determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo `MQQueueManager` no pueden realizar más llamadas IBM MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto `MQQueueManager` .

mensaje
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada `MessageId` y `CorrelationId` estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón `MQRC_BACKED_OUT` después de una reconexión satisfactoria, para los mensajes recibidos en `MQGM_SYNCPOINT`.

Opciones de getMessage
Opciones que controlan la acción de la obtención.

El uso de la opción `MQC.MQGMO_CONVERT` puede dar como resultado una excepción con el código de razón `MQC.MQRC_CONVERTED_STRING_TOO_BIG` al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *`getMessageOptions`* , la opción de mensaje utilizada es `MQGMO_NOWAIT`.

Si utiliza la opción `MQGMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE` .

Tamaño de MaxMsg
El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo `MQGMO_ACCEPT_TRUNCATED_MSG` se establece en el objeto `MQGetMessageOptions` , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación `MQCC_WARNING` y el código de razón `MQRC_TRUNCATED_MSG_ACCEPTED` .
- Si el distintivo `MQGMO_ACCEPT_TRUNCATED_MSG` no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación `MQCC_WARNING` y el código de razón `MQRC_TRUNCATED_MSG_FAILED` .

Si no se especifica *`MaxMsgSize`* , se recupera todo el mensaje.

`public void Put(MQMessage message);`
`public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);`

Genera `MQException`.

Coloca un mensaje en una cola si el destino es un objeto `MQQueue` o publica un mensaje en un tema si el destino es un objeto `MQTopic` .

Las modificaciones en el objeto `MQMessage` después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola IBM MQ .

`Put` actualiza las propiedades `MessageId` y `CorrelationId` del objeto `MQMessage` y no borra los datos del mensaje. Las llamadas `Put` o `Get` adicionales hacen referencia a la información actualizada en el objeto `MQMessage` . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

mensaje
Un objeto `MQMessage` que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- `MQRC_CALL_INTERRUPTED` si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.

- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

Opciones de putMessage

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

Constructores

MQDestination es una clase base abstracta y no se puede instanciar. Acceda a destinos utilizando constructores MQQueue y MQTopic , o utilizando MQQueueManager.AccessQueue y MQQueueManager.AccessTopic methods.

Clase MQEnvironment.NET

Utilice MQEnvironment para controlar cómo se llama al constructor MQQueueManager y para seleccionar una conexión IBM MQ MQI client . La clase MQEnvironment contiene propiedades que controlan el comportamiento del IBM MQ.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```


- “Propiedades-sólo cliente” en la página 1774
- “Propiedades” en la página 1775
- “Constructores” en la página 1776

Propiedades-sólo cliente

Prueba de MQException que se genera al obtener las propiedades.

public static int CertificateValPolicy {get; set;}

Establezca qué política de validación de certificados TLS se utiliza para validar los certificados digitales recibidos de los sistemas asociados remotos. Los valores válidos son:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

public static ArrayList EncryptionPolicySuiteB {get; set;}

Establezca el nivel de criptografía compatible con Suite B. Los valores válidos son:

- MQC.MQ_SUITE_B_NONE -Este es el valor predeterminado.
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}

El nombre del canal para conectarse al gestor de colas de destino. Debe establecer la propiedad de canal antes de crear una instancia de MQQueueManager en modalidad de cliente.

public static int FipsRequired {get; set;}

Especifique MQC.MQSSL_FIPS_YES para utilizar sólo algoritmos certificados por FIPS si el cifrado se lleva a cabo en IBM MQ. El valor predeterminado es MQC.MQSSL_FIPS_NO.

Si el hardware criptográfico está configurado, los módulos criptográficos utilizados son los proporcionados por el producto de hardware. En función del hardware en uso, es posible que no estén certificados por FIPS a un nivel determinado.

public static string Hostname {get; set;}
El nombre de host TCP/IP del sistema en el que reside el servidor IBM MQ . Si no se establece el nombre de host y no se establecen propiedades de alteración temporal, se utiliza la modalidad de enlaces de servidor para conectarse al gestor de colas local.

public static int Port {get; set;}
El puerto al que conectarse. Es el puerto en el que el servidor de IBM MQ está a la escucha de solicitudes de conexión entrantes. El valor predeterminado es 1414.

public static string SSLCipherSpec {get; set;}
Establezca SSLCipherSpec en el valor de CipherSpec establecido en el canal SVRCONN para habilitar TLS para la conexión. El valor predeterminado es Null y TLS no está habilitado para la conexión.

public static string sslPeerName {get; set;}

Un patrón de nombre distinguido. Si se establece sslCipherSpec , esta variable se puede utilizar para asegurarse de que se utiliza el gestor de colas correcto. Si se establece en nulo (valor predeterminado), el DN del gestor de colas no se realiza. sslPeerNombre se ignora si sslCipherSpec es nulo.

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public static ArrayList HdrCompList {get; set;}
Lista de compresión de datos de cabecera

public static int KeyResetCount {get; set;}
Indica el número de bytes no cifrados enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta.

public static ArrayList MQAIRArray {get; set;}
Una matriz de objetos MQAuthenticationInformationRecord .

public static ArrayList MsgCompList {get; set;}
Lista de compresión de datos de mensaje

public static string Password {get; set;}
La contraseña que se va a autenticar. La contraseña a la que se hace referencia desde la estructura MQCSP se rellena estableciendo esta propiedad de contraseña.

public static string ReceiveExit {get; set;}
Una salida de recepción le permite examinar y modificar los datos recibidos de un gestor de colas. Normalmente se utiliza con una salida de emisión correspondiente en el gestor de colas. Si ReceiveExit se establece en nulo, no se llama a ninguna salida de recepción.

public static string ReceiveUserData {get; set;}
Los datos de usuario asociados a una salida de recepción. Limitado a 32 caracteres.

public static string SecurityExit {get; set;}
Una salida de seguridad le permite personalizar los flujos de seguridad que se producen cuando se intenta conectarse a un gestor de colas. Si SecurityExit se establece en nulo, no se llama a ninguna salida de seguridad.

public static string SecurityUserData {get; set;}
Los datos de usuario asociados a una salida de seguridad. Limitado a 32 caracteres.

public static string SendExit {get; set;}

Una salida de emisión le permite examinar o modificar los datos enviados a un gestor de colas. Normalmente se utiliza con una salida de recepción correspondiente en el gestor de colas. Si `SendExit` se establece en nulo, no se llama a ninguna salida de envío.

public static string SendUserData {get; set;}

Los datos de usuario asociados a una salida de emisión. Limitado a 32 caracteres.

public static string SharingConversations {get; set;}

El campo `SharingConversations` se utiliza en conexiones de aplicaciones .NET , cuando estas aplicaciones no utilizan una tabla de definiciones de canal de cliente (CCDT).

`SharingConversations` determina el número máximo de conversaciones que se pueden compartir en un socket asociado a esta conexión.

Un valor de 0 significa que el canal funciona como lo hacía antes de IBM WebSphere MQ 7.0, con respecto a la compartición de conversación, la lectura anticipada y la pulsación.

El campo se pasa en la tabla hash de propiedades como un `SHARING_CONVERSATIONS_PROPERTY`, al crear una instancia de un gestor de colas de IBM MQ .

Si no especifica `SharingConversations`, se utiliza un valor predeterminado de 10.

public static string SSLCryptoHardware {get; set;}

Establece el nombre de la serie de parámetro necesaria para configurar el hardware criptográfico presente en el sistema. `SSLCryptoHardware` se ignora si `sslCipherSpec` es nulo.

public static string SSLKeyRepository {get; set;}

Establezca el nombre de archivo completo del repositorio de claves.

Si `SSLKeyRepository` se establece en nulo (valor predeterminado), se utiliza la variable de entorno `MQSSLKEYR` del certificado para localizar el repositorio de claves. `SSLCryptoHardware` se ignora si `sslCipherSpec` es nulo.

Nota: La extensión `.kdb` es una parte obligatoria del nombre de archivo, pero no se incluye como parte del valor del parámetro. El directorio que especifique debe existir. IBM MQ crea el archivo la primera vez que accede al nuevo repositorio de claves, a menos que el archivo ya exista.

public static string UserId {get; set;}

El ID de usuario que se va a autenticar. El ID de usuario al que se hace referencia desde la estructura `MQCSP` se rellena estableciendo `UserId`. Autentique `UserId` utilizando una salida de API o de seguridad.

Constructores

public MQEnvironment()

Clase `MQException`.NET

Utilice `MQException` para averiguar el código de terminación y de razón de una función IBM MQ anómala. Se genera un `MQException` siempre que se produce un error IBM MQ .

Clase

```
System.Object
        │
        └─ System.Exception
                │
                └─ System.ApplicationException
                        │
                        └─ IBM.WMQ.MQException
```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```


- “Propiedades” en la página 1777
- “Constructores” en la página 1777

Propiedades

public int CompletionCode {get; set;}

El código de terminación de IBM MQ asociado con el error. Los valores posibles son:

- `MQException.MQCC_OK`
- `MQException.MQCC_WARNING`
- `MQException.MQCC_FAILED`

public int ReasonCode {get; set;}

Código de razón IBM MQ que describe el error.

Constructores

public MQException(int *completionCode*, int *reasonCode*)

completionCode

El código de terminación de IBM MQ .

reasonCode

El código de terminación de IBM MQ .

Clase MQGetMessageOptions.NET

Utilice `MQGetMessageOptions` para especificar cómo se recuperan los mensajes. Modifica el comportamiento de `MQDestination.Get`.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```


- “Propiedades” en la página 1777
- “Constructores” en la página 1780

Propiedades

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco *.

Prueba de `MQException` que se genera al obtener las propiedades.

public int GroupStatus {get;}*

`GroupStatus` indica si el mensaje recuperado está en un grupo y si es el último del grupo. Los valores posibles son:

MQC.MQGS_LAST_MSG_IN_GROUP

El mensaje es el último o único mensaje del grupo.

MQC.MQGS_MSG_IN_GROUP

El mensaje está en un grupo, pero no es el último del grupo.

MQC.MQGS_NOT_IN_GROUP
El mensaje no está en un grupo.

public int MatchOptions {get; set;}*
`MatchOptions` determina cómo se selecciona un mensaje. Se pueden establecer las siguientes opciones de coincidencia:

MQC.MQMO_MATCH_CORREL_ID
ID de correlación que debe coincidir.

MQC.MQMO_MATCH_GROUP_ID
ID de grupo que debe coincidir.

MQC.MQMO_MATCH_MSG_ID
ID de mensaje que debe coincidir.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER
Coincide con el número de secuencia de mensaje.

MQC.MQMO_NONE
No se requiere ninguna coincidencia.

public int Options {get; set;}
Las `Opciones` controlan la acción de `MQQueue.get`. Se puede especificar cualquiera de los valores siguientes. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG
Permitir truncamiento de datos de mensaje.

MQC.MQGMO_ALL_MSGS_AVAILABLE*
Recuperar mensajes de un grupo sólo cuando todos los mensajes del grupo están disponibles.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*
Recuperar los segmentos de un mensaje lógico sólo cuando todos los segmentos del grupo están disponibles.

MQC.MQGMO_BROWSE_FIRST
Examinar desde el inicio de la cola.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*
Examine el mensaje bajo el cursor para examinar.

MQC.MQGMO_BROWSE_NEXT
Examinar desde la posición actual en la cola.

MQC.MQGMO_COMPLETE_MSG*
Recuperar sólo mensajes lógicos completos.

MQC.MQGMO_CONVERT
Solicite que los datos de aplicación se conviertan, para que se ajusten a los atributos `CharacterSet` y `Codificación` del `MQMessage`, antes de que los datos se copien en el almacenamiento intermedio de mensajes. Puesto que la conversión de datos también se aplica cuando los datos se recuperan del almacenamiento intermedio de mensajes, las aplicaciones no establecen esta opción.

El uso de esta opción puede causar problemas al convertir de juegos de caracteres de un solo byte a juegos de caracteres de doble byte. En su lugar, realice la conversión utilizando los métodos `readString`, `readLiney` `writeString` después de que se haya entregado el mensaje.

MQC.MQGMO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQGMO_LOCK*
Bloquear el mensaje que se examina.

MQC.MQGMO_LOGICAL_ORDER*
Devuelve mensajes en grupos y segmentos de mensajes lógicos, en orden lógico.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, el código de razón MQRC_RECONNECT_INCOMPATIBLE se devuelve a la aplicación.

MQC.MQGMO_MARK_SKIP_BACKOUT*
Permitir que se restituya una unidad de trabajo sin restablecer el mensaje en la cola.

MQC.MQGMO_MSG_UNDER_CURSOR
Obtener mensaje bajo el cursor para examinar.

MQC.MQGMO_NONE
No se ha especificado ninguna otra opción; todas las opciones asumen sus valores predeterminados.

MQC.MQGMO_NO_PROPERTIES
No se recupera ninguna propiedad del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión).

MQC.MQGMO_NO_SYNCPOINT
Obtener mensaje sin control de punto de sincronización.

MQC.MQGMO_NO_WAIT
Vuelva inmediatamente si no hay ningún mensaje adecuado.

MQC.MQGMO_PROPERTIES_AS_Q_DEF
Recupere las propiedades de mensaje tal como las define el atributo PropertyControl de MQQueue. El acceso a las propiedades del mensaje en el descriptor de mensaje, o extensión, no se ve afectado por el atributo PropertyControl .

MQC.MQGMO_PROPERTIES_COMPATIBILITY
Recupere las propiedades de mensaje con el prefijo mcd, jms, usro mqext, en las cabeceras MQRFH2 . Otras propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, se descartan.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2
Recuperar propiedades de mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, en las cabeceras MQRFH2 . Utilice MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 en aplicaciones que esperan recuperar propiedades pero que no se pueden cambiar para utilizar manejadores de mensajes.

MQC.MQGMO_PROPERTIES_IN_HANDLE
Recuperar propiedades de mensaje utilizando un MsgHandle.

MQC.MQGMO_SYNCPOINT
Obtenga el mensaje bajo control de punto de sincronización. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*
Obtener mensaje con control de punto de sincronización si el mensaje es persistente.

MQC.MQGMO_UNLOCK*
Desbloquear un mensaje bloqueado anteriormente.

MQC.MQGMO_WAIT
Espere a que llegue un mensaje.

public string ResolvedQueueName {get;}
El gestor de colas establece ResolvedQueueNombre en el nombre local de la cola de la que se ha recuperado el mensaje. ResolvedQueueNombre es diferente del nombre utilizado para abrir la cola si se ha abierto una cola alias o una cola modelo.

public char Segmentation {get;}*
Segmentación indica si puede permitir la segmentación para el mensaje recuperado. Los valores posibles son:

MQC.MQSEG_INHIBITED
No permitir segmentación.

MQC.MQSEG_ALLOWED
Permitir segmentación

public byte SegmentStatus {get;}*
`SegmentStatus` es un campo de salida que indica si el mensaje recuperado es un segmento de un mensaje lógico. Si el mensaje es un segmento, el distintivo indica si es el último segmento. Los valores posibles son:

MQC.MQSS_LAST_SEGMENT
El mensaje es el último o único segmento del mensaje lógico.

MQC.MQSS_NOT_A_SEGMENT
El mensaje no es un segmento.

MQC.MQSS_SEGMENT
El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

public int WaitInterval {get; set;}
`WaitInterval` es el tiempo máximo en milisegundos que una llamada `MQQueue.get` espera a que llegue un mensaje adecuado. Utilice `WaitInterval` con `MQC.MQGMO_WAIT`. Establezca un valor de `MQC.MQWI_UNLIMITED` para esperar un tiempo ilimitado para un mensaje.

Constructores

public MQGetMessageOptions()
Construya un nuevo objeto `MQGetMessageOptions` con `Options` establecido en `MQC.MQGMO_NO_WAIT`, `WaitInterval` establecido en cero y `ResolvedQueueName` establecido en blanco.

Clase MQManagedObject.NET

Utilice `MQManagedObject` para consultar y establecer atributos de `MQDestination`, `MQProcess`, `MQQueueManager` y `MQSubscription`. `MQManagedObject` es una superclase de estas clases.

Clases

```
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;`

- “Propiedades” en la página 1780
- “Métodos” en la página 1781
- “Constructores” en la página 1782

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public string AlternateUserId {get; set;}
El ID de usuario alternativo, si lo hay, establecido cuando se abrió el recurso. `AlternateUserID.set` se ignora cuando se emite para un objeto que está abierto. `AlternateUserId` no es válido para suscripciones.

public int CloseOptions {get; set;}

Establezca este atributo para controlar la forma en que se cierra el recurso. El valor predeterminado es `MQC.MQCO_NONE`. `MQC.MQCO_NONE` es el único valor permitido para todos los recursos que no sean colas dinámicas permanentes, colas dinámicas temporales, suscripciones y temas a los que acceden los objetos que las han creado.

Para colas y temas, se permiten los siguientes valores adicionales:

MQC.MQCO_DELETE
Suprima la cola si no hay mensajes.

MQC.MQCO_DELETE_PURGE
Suprima la cola, depurando los mensajes que haya en ella.

MQC.MQCO_QUIESCE
Solicite que se cierre la cola, recibiendo un aviso si queda algún mensaje (permitiendo que se recuperen antes del cierre final).

Para suscripciones, se permiten los siguientes valores adicionales:

MQC.MQCO_KEEP_SUB
La suscripción no se suprime. Esta opción sólo es válida si la suscripción original es duradera. `MQC.MQCO_KEEP_SUB` es el valor predeterminado para un tema duradero.

MQC.MQCO_REMOVE_SUB
La suscripción se suprime. `MQC.MQCO_REMOVE_SUB` es el valor predeterminado para un tema no duradero y no gestionado.

MQC.MQCO_PURGE_SUB
La suscripción se suprime. `MQC.MQCO_PURGE_SUB` es el valor predeterminado para un tema gestionado no duradero.

public MQQueueManager ConnectionReference {get;}
El gestor de colas al que pertenece este recurso.

public string MQDescription {get;}
La descripción del recurso tal como la retiene el gestor de colas. `MQDescription` devuelve una serie vacía para suscripciones y temas.

public boolean IsOpen {get;}
Indica si el recurso está abierto actualmente.

public string Name {get;}
Nombre del recurso. El nombre es el proporcionado en el método de acceso o el asignado por el gestor de colas para una cola dinámica.

public int OpenOptions {get; set;}
Las OpenOptions se establecen cuando se abre un objeto IBM MQ . El método `OpenOptions.set` se ignora y no provoca un error. Las suscripciones no tienen `OpenOptions`.

Métodos

public virtual void Close();
Genera `MQException`.

Cierra el objeto. No se permiten más operaciones en este recurso después de llamar a `Close`. Para cambiar el comportamiento del método `Close` , establezca el atributo `closeOptions` .

public string GetAttributeString(int *selector*, int *length*);
Genera `MQException`.

Obtiene una serie de atributo.

selector
Entero que indica qué atributo se está consultando.

length
Entero que indica la longitud de la serie necesaria.

public void Inquire(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de una cola, un proceso o un gestor de colas. Los atributos que se van a consultar se especifican en la matriz de selectores.

Nota: Muchos de los atributos más comunes se pueden consultar utilizando los métodos `Get` definidos en `MQManagedObject`, `MQQueue` y `MQQueueManager`.

selectores
Matriz de enteros que identifica los atributos con los valores que se van a consultar.

intAttrs
La matriz en la que se devuelven los valores de atributo de entero. Los valores de atributos enteros se devuelven en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs
El almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenado. Los atributos de caracteres se devuelven en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada serie de atributo es fija para cada atributo.

public void Set(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Establece los atributos definidos en el vector de selectores. Los atributos que se van a establecer se especifican en la matriz de selectores.

selectores
Matriz de enteros que identifica los atributos con valores que se van a establecer.

intAttrs
La matriz de valores de atributo de entero que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs
El almacenamiento intermedio en el que se concatenan los atributos de carácter que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada atributo de carácter es fija.

public void SetAttributeString(int *selector*, string *value*, int *length*);
Genera `MQException`.

Establece una serie de atributo.

selector
Entero que indica qué atributo se está definiendo.

valor
La serie que se va a establecer como valor de atributo.

length
Entero que indica la longitud de la serie necesaria.

Constructores

protected MQManagedObject()
Método constructor. Este objeto es una clase base abstracta que no se puede instanciar por sí misma.

Clase MQMessage.NET

Utilice MQMessage para acceder al descriptor de mensaje y a los datos de un mensaje IBM MQ . MQMessage encapsula un mensaje IBM MQ .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Cree un objeto MQMessage y, a continuación, utilice los métodos Read y Write para transferir datos entre el mensaje y otros objetos de la aplicación. Enviar y recibir objetos MQMessage utilizando los métodos Put y Get de las clases MQDestination, MQQueue y MQTopic .

Obtenga y establezca las propiedades del descriptor de mensaje utilizando las propiedades de MQMessage. Establezca y obtenga propiedades de mensajes ampliadas utilizando los métodos SetProperty y GetProperty .

- "Propiedades" en la página 1783
- "Métodos de mensaje Read y Write" en la página 1789
- "Métodos de almacenamiento intermedio" en la página 1792
- "Métodos de propiedad" en la página 1792
- "Constructores" en la página 1795

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public string AccountingToken {get; set;}
: Parte del contexto de identidad del mensaje; ayuda a una aplicación a cobrar por el trabajo realizado como resultado del mensaje. El valor predeterminado es MQC.MQACT_NONE.

public string ApplicationIdData {get; set;}
: Parte del contexto de identidad del mensaje. ApplicationIdDatos es información definida por la suite de aplicaciones y se puede utilizar para proporcionar información adicional sobre el mensaje o su originador. El valor predeterminado es "".

public string ApplicationOriginData {get; set;}
: Información definida por la aplicación que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. El valor predeterminado es "".

public int BackoutCount {get;}
: Recuento del número de veces que una llamada MQQueue.Get ha devuelto y restituido previamente el mensaje como parte de una unidad de trabajo. El valor predeterminado es cero.

public int CharacterSet {get; set;}

El identificador de juego de caracteres codificado de datos de caracteres en el mensaje.

Establezca CharacterSet para identificar el juego de caracteres de los datos de caracteres en el mensaje. Obtenga CharacterSet para averiguar en qué juego de caracteres se ha utilizado para codificar datos de caracteres en el mensaje.

Las aplicaciones .NET siempre se ejecutan en Unicode, mientras que en otros entornos las aplicaciones se ejecutan en el mismo juego de caracteres con el que se ejecuta el gestor de colas.

Los métodos `ReadString` y `ReadLine` convierten los datos de caracteres del mensaje a Unicode automáticamente.

El método `WriteString` se convierte de Unicode al juego de caracteres codificado en `CharacterSet`. Si `CharacterSet` se establece en su valor predeterminado, `MQC.MQCCSI_Q_MGR`, que es 0, no tiene lugar ninguna conversión y `CharacterSet` se establece en 1200. Si establece `CharacterSet` en algún otro valor, `WriteString` se convierte de Unicode al valor alternativo.

Nota: Otros métodos de lectura y escritura no utilizan `CharacterSet`.

- `ReadChar` y `WriteChar` leen y escriben un carácter Unicode en y desde el almacenamiento intermedio de mensajes sin conversión.
- `ReadUTF` y `WriteUTF` se convierten entre una serie Unicode en la aplicación y una serie UTF-8 , con el prefijo de un campo de longitud de 2 bytes, en el almacenamiento intermedio de mensajes.
- Los métodos de bytes transfieren bytes entre la aplicación y el almacenamiento intermedio de mensajes sin modificación.

`public byte[] CorrelationId {get; set;}`

- Para una llamada `MQQueue.Get` , el identificador de correlación del mensaje que se va a recuperar. El gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. El valor predeterminado, `MQC.MQCI_NONE`, ayuda a que coincida cualquier identificador de correlación.
- Para una llamada `MQQueue.Put` , el identificador de correlación a establecer.

`public int DataLength {get;}`
Número de bytes de datos de mensaje que quedan por leer.

`public int DataOffset {get; set;}`
La posición actual del cursor dentro de los datos del mensaje. Las lecturas y grabaciones entran en vigor en la posición actual.

`public int Encoding {get; set;}`
Representación utilizada para valores numéricos en los datos de mensaje de aplicación. `Codificación` se aplica a datos binarios, decimales empaquetados y de coma flotante. El comportamiento de los métodos de lectura y escritura para estos formatos numéricos se modifica en consecuencia. Construya un valor para el campo de codificación añadiendo un valor de cada una de estas tres secciones. De forma alternativa, construya el valor que combina los valores de cada una de las tres secciones utilizando el operador OR a nivel de bit.

1. Entero binario

   `MQC.MQENC_INTEGER_NORMAL`
   Enteros big-endian.

   `MQC.MQENC_INTEGER_REVERSED`
   Enteros little-endian, tal como se utiliza en la arquitectura de Intel .

2. Decimal empaquetado

   `MQC.MQENC_DECIMAL_NORMAL`
   Big-endian packed-decimal, tal como lo utiliza z/OS.

   `MQC.MQENC_DECIMAL_REVERSED`
   Decimal empaquetado little-endian.

3. Coma flotante

   `MQC.MQENC_FLOAT_IEEE_NORMAL`
   Flotadores IEEE de big-endian.

   `MQC.MQENC_FLOAT_IEEE_REVERSED`
   Little-endian IEEE floats, tal como se utiliza en la arquitectura de Intel .

   `MQC.MQENC_FLOAT_S390`
   z/OS formatear puntos flotantes.

El valor predeterminado es:

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

El valor predeterminado hace que `WriteInt` escriba un entero little-endian y que `ReadInt` lea un entero little-endian. Si establece el distintivo `MQC.MQENC_INTEGER_NORMAL` en su lugar, `WriteInt` escribe un entero big-endian y `ReadInt` lee un entero big-endian.

Nota: Se puede producir una pérdida de precisión al convertir de puntos flotantes de formato IEEE a puntos flotantes de formato zSeries .

public int Expiry {get; set;}
Tiempo de caducidad expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. Una vez transcurrido el tiempo de caducidad de un mensaje, el gestor de colas lo puede descartar. Si el mensaje ha especificado uno de los distintivos `MQC.MQRO_EXPIRATION` , se genera un informe cuando se descarta el mensaje. El valor predeterminado es `MQC.MQEI_UNLIMITED`, lo que significa que el mensaje nunca caduca.

public int Feedback {get; set;}
Utilice `Comentarios` con un mensaje de tipo `MQC.MQMT_REPORT` para indicar la naturaleza del informe. El sistema define los siguientes códigos de retorno:

- `MQC.MQFB_EXPIRATION`
- `MQC.MQFB_COA`
- `MQC.MQFB_COD`
- `MQC.MQFB_QUIT`
- `MQC.MQFB_PAN`
- `MQC.MQFB_NAN`
- `MQC.MQFB_DATA_LENGTH_ZERO`
- `MQC.MQFB_DATA_LENGTH_NEGATIVE`
- `MQC.MQFB_DATA_LENGTH_TOO_BIG`
- `MQC.MQFB_BUFFER_OVERFLOW`
- `MQC.MQFB_LENGTH_OFF_BY_ONE`
- `MQC.MQFB_IIH_ERROR`

También se pueden utilizar los valores de comentarios definidos por la aplicación en el rango de `MQC.MQFB_APPL_FIRST` a `MQC.MQFB_APPL_LAST` . El valor predeterminado de este campo es `MQC.MQFB_NONE`, lo que indica que no se proporciona ningún comentario.

public string Format {get; set;}
Nombre de formato utilizado por el remitente del mensaje para indicar al destinatario la naturaleza de los datos del mensaje. Puede utilizar sus propios nombres de formato, pero los nombres que empiezan por las letras `MQ` tienen significados definidos por el gestor de colas. Los formatos incorporados del gestor de colas son:

MQC.MQFMT_ADMIN
Mensaje de solicitud/respuesta del servidor de mandatos.

MQC.MQFMT_COMMAND_1
Mensaje de respuesta de mandato de tipo 1.

MQC.MQFMT_COMMAND_2
Mensaje de respuesta de mandato de tipo 2.

MQC.MQFMT_DEAD_LETTER_HEADER
Cabecera de mensaje no entregado.

MQC.MQFMT_EVENT
Mensaje de suceso.

MQC.MQFMT_NONE
No hay nombre de formato.

MQC.MQFMT_PCF
Mensaje definido por el usuario en formato de mandato programable.

MQC.MQFMT_STRING
Mensaje que consta totalmente de caracteres.

MQC.MQFMT_TRIGGER
Mensaje de desencadenante

MQC.MQFMT_XMIT_Q_HEADER
Cabecera de cola de transmisión.

El valor predeterminado es `MQC.MQFMT_NONE`.

public byte[] GroupId {get; set;}
Serie de bytes que identifica el grupo de mensajes al que pertenece el mensaje físico. El valor predeterminado es `MQC.MQGI_NONE`.

public int MessageFlags {get; set;}
Distintivos que controlan la segmentación y el estado de un mensaje.

public byte[] MessageId {get; set;}

Para una llamada `MQQueue.Get` , este campo especifica el identificador de mensaje del mensaje que se va a recuperar. Normalmente, el gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. Permitir que cualquier identificador de mensaje coincida utilizando el valor especial `MQC.MQMI_NONE`.

Para una llamada `MQQueue.Put` , este campo especifica el identificador de mensaje que se va a utilizar. Si se especifica `MQC.MQMI_NONE` , el gestor de colas genera un identificador de mensaje exclusivo cuando se coloca el mensaje. El valor de esta variable de miembro se actualiza después de la colocación, para indicar el identificador de mensaje que se ha utilizado. El valor predeterminado es `MQC.MQMI_NONE`.

public int MessageLength {get;}
Número de bytes de datos de mensaje en el objeto `MQMessage` .

public int MessageSequenceNumber {get; set;}
Número de secuencia de un mensaje lógico dentro de un grupo.

public int MessageType {get; set;}
Indica el tipo del mensaje. El sistema define actualmente los valores siguientes:

- `MQC.MQMT_DATAGRAM`
- `MQC.MQMT_REPLY`
- `MQC.MQMT_REPORT`
- `MQC.MQMT_REQUEST`

También se pueden utilizar valores definidos por la aplicación, en el rango de `MQC.MQMT_APPL_FIRST` a `MQC.MQMT_APPL_LAST`. El valor predeterminado de este campo es `MQC.MQMT_DATAGRAM`.

public int Offset {get; set;}
En un mensaje segmentado, el desplazamiento de datos en un mensaje físico desde el inicio de un mensaje lógico.

public int OriginalLength {get; set;}
La longitud original de un mensaje segmentado.

public int Persistence {get; set;}

Persistencia de mensajes. Los valores siguientes están definidos:

- `MQC.MQPER_NOT_PERSISTENT`

Si establece esta opción en un cliente reconectable, el código de razón MQRC_NONE se devuelve a la aplicación cuando la conexión es satisfactoria.

- MQC.MQPER_PERSISTENT

  Si establece esta opción en un cliente reconectable, el código de razón MQRC_CALL_INTERRUPTED se devuelve a la aplicación después de que la conexión sea satisfactoria.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

El valor predeterminado es MQC.MQPER_PERSISTENCE_AS_Q_DEF, que toma la persistencia del mensaje del atributo de persistencia predeterminado de la cola de destino.

public int Priority {get; set;}
Prioridad del mensaje. El valor especial MQC.MQPRI_PRIORITY_AS_Q_DEF también se puede establecer en el mensaje de salida. A continuación, la prioridad del mensaje se toma del atributo de prioridad predeterminado de la cola de destino. El valor predeterminado es MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}
Especifica si la validación de propiedades tiene lugar cuando se establece una propiedad del mensaje. Los valores posibles son:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

El valor predeterminado es MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}
El nombre de la aplicación que ha transferido el mensaje. El valor predeterminado es "".

public int PutApplicationType {get; set;}
El tipo de aplicación que transfiere el mensaje. PutApplicationTipo puede ser un valor definido por el sistema o definido por el usuario. El sistema define los valores siguientes:

- MQC.MQAT_AIX
- MQC.MQAT_CICS
- MQC.MQAT_DOS
- MQC.MQAT_IMS
- MQC.MQAT_MVS
- MQC.MQAT_OS2
- MQC.MQAT_OS400
- MQC.MQAT_QMGR
- MQC.MQAT_UNIX
- MQC.MQAT_WINDOWS
- MQC.MQAT_JAVA

El valor predeterminado es MQC.MQAT_NO_CONTEXT, que indica que no hay información de contexto en el mensaje.

public DateTime PutDateTime {get; set;}
La fecha y hora en que se ha colocado el mensaje.

public string ReplyToQueueManagerName {get; set;}
El nombre del gestor de colas para enviar mensajes de respuesta o de informe. El valor predeterminado es ""y el gestor de colas proporciona el ReplyToQueueManagerName.

public string ReplyToQueueName {get; set;}
El nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud get para el mensaje envía mensajes MQC.MQMT_REPLY y MQC.MQMT_REPORT . El valor predeterminado de ReplyToQueueName es "".

public int Report {get; set;}

Utilice `Informe` para especificar opciones sobre los mensajes de informe y respuesta:

- Indica si los informes son necesarios.
- Indica si los datos de mensaje de aplicación se van a incluir en los informes.
- Cómo establecer los identificadores de mensaje y correlación en el informe o respuesta.

Se puede solicitar cualquier combinación de los cuatro tipos de informe:

- Especifique cualquier combinación de los cuatro tipos de informe. Seleccionar cualquiera de las tres opciones para cada tipo de informe, en función de si los datos del mensaje de aplicación se van a incluir en el mensaje de informe.
  1. Confirmar al recibir
     - `MQC.MQRO_COA`
     - `MQC.MQRO_COA_WITH_DATA`
     - `MQC.MQRO_COA_WITH_FULL_DATA` **
  2. Confirmar al entregar
     - `MQC.MQRO_COD`
     - `MQC.MQRO_COD_WITH_DATA`
     - `MQC.MQRO_COD_WITH_FULL_DATA` **
  3. Excepción
     - `MQC.MQRO_EXCEPTION`
     - `MQC.MQRO_EXCEPTION_WITH_DATA`
     - `MQC.MQRO_EXCEPTION_WITH_FULL_DATA` **
  4. Caducidad
     - `MQC.MQRO_EXPIRATION`
     - `MQC.MQRO_EXPIRATION_WITH_DATA`
     - `MQC.MQRO_EXPIRATION_WITH_FULL_DATA` **

  Nota: Los valores marcados con ** en la lista no están soportados por los gestores de colas de z/OS . No los utilice si es probable que la aplicación acceda a un gestor de colas de z/OS , independientemente de la plataforma en la que se ejecute la aplicación.
- Especifique una de las opciones siguientes para controlar cómo se genera el ID de mensaje para el informe o el mensaje de respuesta:
  - `MQC.MQRO_NEW_MSG_ID`
  - `MQC.MQRO_PASS_MSG_ID`
- Especifique una de las opciones siguientes para controlar cómo se va a establecer el ID de correlación del informe o mensaje de respuesta:
  - `MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID`
  - `MQC.MQRO_PASS_CORREL_ID`
- Especifique uno de los siguientes para controlar la disposición del mensaje original cuando no se pueda entregar a la cola de destino:
  - `MQC.MQRO_DEAD_LETTER_Q`
  - `MQC.MQRO_DISCARD_MSG` **
- Si no se especifica ninguna opción de informe, el valor predeterminado es:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- Puede especificar uno o ambos de los siguientes para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.
  - MQC.MQRO_PAN
  - MQC.MQRO_NAN

public int TotalMessageLength {get;}
Número total de bytes en el mensaje tal como se ha almacenado en la cola de mensajes desde la que se ha recibido este mensaje.

public string UserId {get; set;}
UserId forma parte del contexto de identidad del mensaje. El gestor de colas generalmente proporciona el valor. Puede alterar temporalmente el valor si tiene autorización para establecer el contexto de identidad.

public int Version {get; set;}
La versión de la estructura MQMD en uso.

Métodos de mensaje Read y Write

Los métodos Read y Write realizan las mismas funciones que los miembros de las clases BinaryReader y BinaryWriter en el espacio de nombres .NET System.IO . Consulte MSDN para ver los ejemplos de uso y sintaxis de lenguaje completo. Los métodos leen o escriben desde la posición actual en el almacenamiento intermedio de mensajes. Mueven la posición actual hacia adelante por el número de bytes leídos o grabados.

Nota: Si los datos del mensaje contienen una cabecera MQRFH o MQRFH2 , debe utilizar el método ReadBytes para leer los datos.

- Todos los métodos emiten IOException.
- Los métodos ReadFully redimensionan automáticamente la matriz byte o sbyte de destino para que se ajuste exactamente al mensaje. También se cambia el tamaño de una matriz nula.
- Read métodos throw EndOfStreamException.
- WriteDecimal métodos throw MQException.
- Los métodos ReadString, ReadLine y WriteString se convierten entre Unicode y el juego de caracteres del mensaje; consulte CharacterSet .
- Los métodos Decimal leen y escriben números decimales empaquetados codificados en formato big-endian, MQC.MQENC_DECIMAL_NORMALo little-endian MQC.MQENC_DECIMAL_REVERSE , según el valor de Codificación. Los rangos decimales y los tipos .NET correspondientes son los siguientes:

  Decimal2/short
  -999 a 999

  Decimal4/int
  -9999999 a 9999999

  Decimal8/long
  -999999999999999 a 99999999999999999

- Los métodos Double y Float leen y escriben valores flotantes codificados en formatos big-endian y little-endian de IEE, MQC.MQENC_FLOAT_IEEE_NORMAL y MQC.MQENC_FLOAT_IEEE_REVERSED, o en formato S/390 , MQC.MQENC_FLOAT_S390, según el valor de Codificación.
- Los métodos Int leen y escriben valores enteros codificados en formato big-endian, MQC.MQENC_INTEGER_NORMALo little-endian, MQC.MQENC_INTEGER_REVERSED, según el valor de Codificación. Los enteros están todos con signo, excepto la adición de un tipo entero de 2 bytes sin signo. Los tamaños enteros y los tipos .NET y IBM MQ son los siguientes:

  2 bytes
  short, Int2, ushort, UInt2

  4 bytes
  int, Int4

8 bytes
`long`, `Int8`

- `WriteObject` transfiere la clase de un objeto, los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos, al almacenamiento intermedio de mensajes.
- `ReadObject` crea un objeto a partir de la clase del objeto, la firma de la clase y los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos.

Tabla 843. Métodos de lectura y escritura de mensajes

Tipo de destino	Signaturas de método
Boolean	`public bool ReadBoolean();` `public void WriteBoolean(bool value);`

Byte	`public byte ReadByte()`
	`public byte ReadUnsignedByte()`
	`public void Write(int value)`
	`public void WriteByte(int value)`
	`public void WriteByte(byte value)`
	`public void WriteByte(sbyte value)`

Bytes	`public byte[] ReadBytes(int count)`
	`public void ReadFully(ref byte[] value)`
	`public void ReadFully(ref sbyte[] value)`
	`public void ReadFully(ref byte[] value, int offset,int length)`
	`public void ReadFully(ref sbyte[] value, int offset,int length)`
	`public void Write(byte[] value)`
	`public void Write(sbyte[] value)`
	`public void Write(byte[] value, int offset,int length)`
	`public void Write(sbyte[] value, int offset,int length)`
	`public void WriteBytes(string value)`

Decimal2	`public void WriteDecimal2(short value)`
Decimal4	`public void WriteDecimal4(short value)`
Decimal8	`public void WriteDecimal8(short value)`
Double	`public double ReadDouble()` `public void WriteDouble(double value)`

Tabla 843. Métodos de lectura y escritura de mensajes (continuación)	
Tipo de destino	**Signaturas de método**
Float	`public float ReadFloat()` `public void WriteFloat(float value)`
Int2	`public void WriteInt2(int value)`

Int4	`public int readDecimal4()`
	`public int ReadInt()`
	`public int ReadInt4()`
	`public void WriteInt(int value)`
	`public void WriteInt4(int value)`

Int8	`public void WriteInt8(long value)`

Long	`public long ReadDecimal8()`
	`public long ReadLong()`
	`public long ReadInt8()`
	`public void WriteLong(long value)`

Object	`public Object ReadObject()` `public void WriteObject(Object object)`

Short	`public short ReadShort()`
	`public short ReadDecimal2()`
	`public short ReadInt2()`
	`public void WriteShort(int value)`

string	`public string ReadString(int length)` `public void WriteString(string string)`
Unsigned Short	`public ushort ReadUnsignedShort()` `public ushort ReadUInt2()`

Tabla 843. Métodos de lectura y escritura de mensajes (continuación)	
Tipo de destino	**Signaturas de método**

Unicode	`public string ReadLine()`
	`public char ReadChar()`
	
	`public void WriteChar(int value)`
	`public void WriteChars(string string)`

UTF	`public string ReadUTF()`
	
	`public void WriteUTF(string string)`

Métodos de almacenamiento intermedio

`public void ClearMessage();`

Genera `IOException`.

Descarta los datos del almacenamiento intermedio de mensajes y vuelve a establecer el desplazamiento de datos en cero.

`public void ResizeBuffer(int size)`

Genera `IOException`.

Una sugerencia para el objeto `MQMessage` sobre el tamaño del almacenamiento intermedio que puede ser necesario para las operaciones get posteriores. Si el mensaje contiene actualmente datos de mensaje y el nuevo tamaño es menor que el tamaño actual, los datos del mensaje se truncan.

`public void Seek(int pos)`

Genera `IOException`, `ArgumentOutOfRangeException`, `ArgumentException`.

Mueve el cursor a la posición absoluta en el almacenamiento intermedio de mensajes proporcionado por *pos*. Las lecturas y grabaciones posteriores actúan en esta posición del almacenamiento intermedio.

`public int SkipBytes(int i)`

Genera `IOException`, `EndOfStreamException`.

Avanza n bytes en el almacenamiento intermedio de mensajes y devuelve n, el número de bytes omitidos.

El método `SkipBytes` se bloquea hasta que se produce uno de los sucesos siguientes:

- Se omiten todos los bytes
- Se ha detectado el final del almacenamiento intermedio de mensajes
- Se genera una excepción

Métodos de propiedad

`public void DeleteProperty(string name);`

Genera `MQException`.

Suprime una propiedad con el nombre especificado del mensaje.

nombre
El nombre de la propiedad que se va a suprimir.

public System.Collections.IEnumerator GetPropertyNames(string *name*)

Genera `MQException`.

Devuelve un `IEnumerator` de todos los nombres de propiedad que coinciden con el nombre especificado. El signo de porcentaje `'%'` se puede utilizar al final del nombre como carácter comodín para filtrar las propiedades del mensaje, coincidiendo con cero o más caracteres, incluido el punto.

nombre
El nombre de la propiedad en la que debe coincidir.

Métodos SetProperty y GetProperty

Todos los métodos `SetProperty` y `GetProperty` emiten `MQException`.

El método `SetProperty` de la clase `MQMessage` .NET añade una propiedad nueva si todavía no existe una propiedad. Sin embargo, si la propiedad ya existe, el valor de propiedad proporcionado se añade al final de la lista. Cuando varios valores se establecen en un nombre de propiedad utilizando `SetProperty`, al llamar a `GetProperty` para ese nombre se devuelven estos valores secuencialmente en el orden en el que se han establecido esos valores.

El comportamiento es el mismo para todos los métodos con tipo `Set*Property` y `Get*Property` como `GetLongProperty`, `SetLongProperty`, `GetBooleanProperty`, `SetBooleanProperty`, `GetStringPropertyy` `SetStringProperty`.

Tabla 844. Métodos `SetProperty` y `GetProperty`

Tipo	Signaturas de método

Boolean	`public boolean GetBooleanProperty(string name);`
	`public boolean GetBooleanProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBooleanProperty(string name, boolean value);`
	`public void SetBooleanProperty(string name, MQPropertyDescriptor pd, boolean value);`

Byte	`public sbyte GetByteProperty(string name);`
	`public sbyte GetByteProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetByteProperty(string name, sbyte value);`
	`public void SetByteProperty(string name, MQPropertyDescriptor pd, sbyte value);`

Bytes	`public sbyte[] GetBytesProperty(string name);`
	`public sbyte[] GetBytesProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBytesProperty(string name, sbyte[] value);`
	`public void SetBytesProperty(string name, MQPropertyDescriptor pd, sbyte[] value);`

Double	`public double GetDoubleProperty(string name);`
	`public double GetDoubleProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetDoubleProperty(string name, double value);`
	`public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);`

Tabla 844. Métodos `SetProperty` y `GetProperty` (continuación)	
Tipo	**Signaturas de método**

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);`

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);`

Tabla 844. Métodos `SetProperty` y `GetProperty` (continuación)	
Tipo	**Signaturas de método**

string	`public string GetStringProperty(string name);`
	`public string GetStringProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetStringProperty(string name, string value);`
	`public void SetStringProperty(string name, MQPropertyDescriptor pd, string value);`

Constructores

`public MQMessage();`

Crea un objeto `MQMessage` con información de descriptor de mensaje predeterminado y un almacenamiento intermedio de mensajes vacío.

Clase `MQProcess`.NET

Utilice `MQProcess` para consultar los atributos de un proceso de IBM MQ . Cree un objeto `MQProcess` utilizando un constructor o un método `MQQueueManager` `AccessProcess` .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQProcess
```

`public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;`

- "Propiedades" en la página 1795
- "Constructores" en la página 1796

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public string ApplicationId {get;}`

Obtiene la serie de caracteres que identifica la aplicación que se va a iniciar. `ApplicationId` lo utiliza una aplicación de supervisor desencadenante. `ApplicationId` se envía a la cola de inicio como parte del mensaje desencadenante.

El valor por omisión es nulo.

`public int ApplicationType {get;}`

Identifica el tipo de proceso que debe iniciar una aplicación de supervisor desencadenante. Los tipos estándar están definidos, pero se pueden utilizar otros:

- `MQAT_AIX`
- `MQAT_CICS`
- `MQAT_IMS`
- `MQAT_MVS`

- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

El valor predeterminado es MQAT_NATIVE.

public string EnvironmentData {get;}

Obtiene información sobre el entorno de la aplicación que se va a iniciar.

El valor por omisión es nulo.

public string UserData {get;}

Obtiene información que el usuario ha proporcionado sobre la aplicación que se va a iniciar.

El valor por omisión es nulo.

Constructores

public MQProcess(MQQueueManager *queueManager*, string *processName*, int *openOptions*);
public MQProcess(MQQueueManager *qMgr*, string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera MQException.

Acceda a un proceso IBM MQ en el gestor de colas *qMgr* para consultar los atributos de proceso.

qMgr
Gestor de colas al que acceder.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

AlternateUserid

Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro **openOptions** , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY .

public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*);
public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera `MQException`.

Acceda a un proceso de IBM MQ en este gestor de colas para consultar los atributos de proceso.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- `MQC.MQOO_FAIL_IF_QUIESCING`
- `MQC.MQOO_INQUIRE`
- `MQC.MQOO_SET`
- `MQC.MQOO_ALTERNATE_USER_AUTHORITY`

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

AlternateUserid

Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro **openOptions** , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica `MQOO_ALTERNATE_USER_AUTHORITY` , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` .

Clase MQPropertyDescriptor.NET

Utilice `MQPropertyDescriptor` como parámetro para los métodos `MQMessage` `GetProperty` y `SetProperty` . `MQPropertyDescriptor` describe una propiedad `MQMessage` .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQPropertyDescriptor
```

`public class IBM.WMQ.MQPropertyDescriptor extends System.Object;`

- “Propiedades” en la página 1797
- “Constructores” en la página 1799

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public int Context {get; set;}

El contexto de mensaje al que pertenece la propiedad. Los valores posibles son:

MQC.MQPD_NO_CONTEXT
La propiedad no está asociada a un contexto de mensaje.

MQC.MQPD_USER_CONTEXT
La propiedad está asociada al contexto de usuario.

Si el usuario está autorizado, se guarda una propiedad asociada con el contexto de usuario cuando se recupera un mensaje. Un método Put posterior que haga referencia al contexto guardado, puede pasar la propiedad al nuevo mensaje.

public int CopyOptions {get; set;}

CopyOptions describe en qué tipo de mensaje se puede copiar la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida IBM MQ que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo CopyOptions .

Se puede especificar cualquier combinación de las opciones siguientes. Combine las opciones añadiendo los valores o utilizando ORa nivel de bit.

MQC.MQCOPY_ALL
La propiedad se copia en todos los tipos de mensajes posteriores.

MQC.MQCOPY_FORWARD
La propiedad se copia en un mensaje que se está reenviando.

MQC.MQCOPY_PUBLISH
La propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQC.MQCOPY_REPLY
La propiedad se copia en un mensaje de respuesta.

MQC.MQCOPY_REPORT
La propiedad se copia en un mensaje de informe.

MQC.MQCOPY_DEFAULT
El valor ha indicado que no se ha especificado ninguna otra opción de copia. No existe ninguna relación entre la propiedad y los mensajes posteriores. MQC.MQCOPY_DEFAULT siempre se devuelve para las propiedades del descriptor de mensaje.

MQC.MQCOPY_NONE
Lo mismo que MQC.MQCOPY_DEFAULT

public int Options { set; }
Opciones toma como valor predeterminado CMQC.MQPD_NONE. No puede establecer ningún otro valor.

public int Support { get; set; }

Establezca Soporte para especificar el nivel de soporte necesario para las propiedades de mensaje definidas por IBM MQ. El soporte para todas las demás propiedades es opcional. Se puede especificar cualquiera o ninguno de los valores siguientes

MQC.MQPD_SUPPORT_OPTIONAL
Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas por IBM MQ .

MQC.MQPD_SUPPORT_REQUIRED
Se necesita soporte para la propiedad. Si coloca el mensaje en un gestor de colas que no da soporte a la propiedad definida por IBM MQ, el método falla. Devuelve el código de terminación MQC.MQCC_FAILED y el código de razón MQC.MQRC_UNSUPPORTED_PROPERTY.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Se necesita soporte para la propiedad, si el mensaje está destinado a una cola local. Si coloca el mensaje en una cola local en un gestor de colas que no da soporte a la propiedad definida por IBM MQ, el método falla. Devuelve el código de terminación MQC.MQCC_FAILED y el código de razón MQC.MQRC_UNSUPPORTED_PROPERTY.

No se realiza ninguna comprobación si el mensaje se coloca en un gestor de colas remoto.

Constructores

PropertyDescriptor();
Cree un descriptor de propiedad.

Clase MQPutMessageOptions.NET

Utilice MQPutMessageOptions para especificar cómo se envían los mensajes. Modifica el comportamiento de MQDestination.Put.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- “Propiedades” en la página 1799 “Constructores” en la página 1801

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco, *.

public MQQueue ContextReference {get; set;}

Si el campo options incluye MQC.MQPMO_PASS_IDENTITY_CONTEXT o MQC.MQPMO_PASS_ALL_CONTEXT, establezca este campo para que haga referencia al MQQueue del que se tomará la información de contexto.

El valor inicial de este campo es nulo.

public int InvalidDestCount {get;} *

Generalmente, se utiliza para las listas de distribución, InvalidDest indica el número de mensajes que no se han podido enviar a las colas de una lista de distribución. El recuento incluye las colas que no se han podido abrir y también las colas que se han abierto correctamente, pero para las que la operación de colocación ha fallado.

.NET no da soporte a listas de distribución, pero InvalidDestRecuento se establece al abrir una sola cola.

public int KnownDestCount {get;} *

Generalmente se utiliza para listas de distribución, KnownDestKnownDest indica el número de mensajes que la llamada actual ha enviado correctamente a las colas que se resuelven en las colas locales.

.NET no da soporte a listas de distribución, pero InvalidDestRecuento se establece al abrir una sola cola.

public int Options {get; set;}

Opciones que controlan la acción de MQDestination.put y MQQueueManager.put. Se puede especificar cualquiera de los valores siguientes o ninguno de ellos. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

MQC.MQPMO_ASYNC_RESPONSE
Esta opción hace que la llamada `MQDestination.put` se realice de forma asíncrona, con algunos datos de respuesta.

MQC.MQPMO_DEFAULT_CONTEXT
Asocie el contexto predeterminado con el mensaje.

MQC.MQPMO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQPMO_LOGICAL_ORDER *
Coloque los mensajes lógicos y los segmentos de los grupos de mensajes en su orden lógico.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, el código de razón `MQRC_RECONNECT_INCOMPATIBLE` se devuelve a la aplicación.

MQC.MQPMO_NEW_CORREL_ID *
Genere un nuevo ID de correlación para cada mensaje enviado.

MQC.MQPMO_NEW_MSG_ID *
Genere un nuevo ID de mensaje para cada mensaje enviado.

MQC.MQPMO_NONE
No se ha especificado ninguna opción. No lo utilice con otras opciones.

MQC.MQPMO_NO_CONTEXT
No se debe asociar ningún contexto con el mensaje.

MQC.MQPMO_NO_SYNCPOINT
Colocar un mensaje sin control de punto de sincronización. Si no se especifica la opción de control de punto de sincronización, se presupone un valor predeterminado de ningún punto de sincronización.

MQC.MQPMO_PASS_ALL_CONTEXT
Pasar todo el contexto desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_PASS_IDENTITY_CONTEXT
Pasar contexto de identidad desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_RESPONSE_AS_Q_DEF

Para una llamada `MQDestination.put` , esta opción toma el tipo de respuesta put del atributo `DEFPRESP` de la cola.

Para una llamada `MQQueueManager.put` , esta opción hace que la llamada se realice de forma síncrona.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF
`MQC.MQPMO_RESPONSE_AS_TOPIC_DEF` es un sinónimo de `MQC.MQPMO_RESPONSE_AS_Q_DEF` para su uso con objetos de tema.

MQC.MQPMO_RETAIN

El gestor de colas debe retener la publicación que se está enviando. Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con `MQC.MQRC_PUT_NOT_RETAINED`.

Solicite una copia de esta publicación después de la hora en que se publicó, llamando al método `MQSubscription.RequestPublicationUpdate` . La publicación guardada se envía a las aplicaciones que crean una suscripción sin establecer la opción `MQC.MQSO_NEW_PUBLICATIONS_ONLY` . Compruebe la propiedad de mensaje `MQIsRetained` de una publicación, cuando se reciba, para averiguar si era la publicación retenida.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en la serie de tema. Si hay varias publicaciones retenidas en el árbol de temas que coinciden con la suscripción, se envían todas.

MQC.MQPMO_SET_ALL_CONTEXT
Establezca todo el contexto de la aplicación.

MQC.MQPMO_SET_IDENTITY_CONTEXT
Establezca el contexto de identidad de la aplicación.

MQC.MQPMO_SYNC_RESPONSE
Esta opción hace que la llamada `MQDestination.put` o `MQQueueManager.put` se realice de forma síncrona, con datos de respuesta completos.

MQC.MQPMO_SUPPRESS_REPLYTO
Cualquier información rellenada en los campos `ReplyToQueueName` y `ReplyToQueueManagerName` de la publicación no se pasa a los suscriptores. Si esta opción se utiliza en combinación con una opción de informe que requiere un `ReplyToQueueName`, la llamada falla con `MQC.MQRC_MISSING_REPLY_TO_Q`.

MQC.MQPMO_SYNCPOINT
Colocar un mensaje con control de punto de sincronización. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

`public int RecordFields {get; set;}`*

Información sobre listas de distribución. Las listas de distribución no son compatibles con .NET.

`public string ResolvedQueueManagerName {get;}`

Campo de salida establecido por el gestor de colas en el nombre del gestor de colas que es propietario de la cola especificada por el nombre de cola remota. `ResolvedQueueManagerName` puede ser diferente del nombre del gestor de colas desde el que se ha accedido a la cola si la cola es una cola remota.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

`public string ResolvedQueueName {get;}`

Campo de salida establecido por el gestor de colas en el nombre de la cola en la que se coloca el mensaje. `ResolvedQueueNombre` puede ser diferente del nombre utilizado para abrir la cola si la cola abierta era un alias o una cola modelo.

Sólo se devuelve un valor que no esté en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

`public int UnknownDestCount {get;}`*

Generalmente se utiliza para listas de distribución, `UnknownDestdesconocido` es un campo de salida establecido por el gestor de colas. Informa del número de mensajes que la llamada actual ha enviado satisfactoriamente a las colas que se resuelven en colas remotas.

.NET no da soporte a listas de distribución, pero `InvalidDestRecuento` se establece al abrir una sola cola.

Constructores

`public MQPutMessageOptions();`

Construya un nuevo objeto `MQPutMessageOptions` sin opciones establecidas y un `ResolvedQueueNombre` y `ResolvedQueueManagerNameen` blanco.

Clase MQQueue.NET

Utilice `MQQueue` para enviar y recibir mensajes y atributos de consulta de una cola IBM MQ . Cree un objeto `MQQueue` utilizando un constructor o un método `MQQueueManager.AccessProcess` .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQDestination
                                        │
                                        └─ IBM.WMQ.MQQueue
```

`public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;`

- “Propiedades” en la página 1802
- “Métodos” en la página 1804
- “Constructores” en la página 1806

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public int ClusterWorkLoadPriority {get;}`
Especifica la prioridad de la cola. Este parámetro sólo es válido para colas locales, remotas y alias.

`public int ClusterWorkLoadRank {get;}`
Especifica el rango de la cola. Este parámetro sólo es válido para colas locales, remotas y alias.

`public int ClusterWorkLoadUseQ {get;}`
Especifica el comportamiento de una operación MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remoto. Este parámetro no se aplica si MQPUT se origina en un canal de clúster. Este parámetro sólo es válido para colas locales.

`public DateTime CreationDateTime {get;}`
Fecha y hora en que se ha creado esta cola.

`public int CurrentDepth {get;}`
Obtiene el número de mensajes que hay actualmente en la cola. Este valor se incrementa durante una llamada put y durante la restitución de una llamada get. Se decrementa durante una operación get de no examinar y durante la restitución de una llamada put.

`public int DefinitionType {get;}`
Cómo se ha definido la cola. Los valores posibles son:

- `MQC.MQQDT_PREDEFINED`
- `MQC.MQQDT_PERMANENT_DYNAMIC`
- `MQC.MQQDT_TEMPORARY_DYNAMIC`

`public int InhibitGet {get; set;}`
Controla si puede obtener mensajes en esta cola o para este tema. Los valores posibles son:

- `MQC.MQQA_GET_INHIBITED`
- `MQC.MQQA_GET_ALLOWED`

`public int InhibitPut {get; set;}`
Controla si puede colocar mensajes en esta cola o para este tema. Los valores posibles son:

- `MQQA_PUT_INHIBITED`
- `MQQA_PUT_ALLOWED`

`public int MaximumDepth {get;}`
Número máximo de mensajes que pueden existir en la cola en cualquier momento. Un intento de colocar un mensaje en una cola que ya contiene este número de mensajes falla con el código de razón `MQC.MQRC_Q_FULL`.

public int MaximumMessageLength {get;}
Longitud máxima de los datos de aplicación que pueden existir en cada mensaje de esta cola. Un intento de colocar un mensaje mayor que este valor falla con el código de razón `MQC.MQRC_MSG_TOO_BIG_FOR_Q`.

public int NonPersistentMessageClass {get;}
El nivel de fiabilidad para los mensajes no permanentes colocados en esta cola.

public int OpenInputCount {get;}
Número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola. `OpenInput` es el número total de descriptores de contexto de entrada válidos conocidos por el gestor de colas local, no sólo los descriptores de contexto creados por la aplicación.

public int OpenOutputCount {get;}
Número de descriptores de contexto válidos actualmente para añadir mensajes a la cola. `OpenOutput` es el número total de descriptores de contexto de salida válidos conocidos por el gestor de colas local, no sólo los descriptores de contexto creados por la aplicación.

public int QueueAccounting {get;}
Especifica si puede habilitar la recopilación de información de contabilidad para la cola.

public int QueueMonitoring {get;}
Especifica si puede habilitar la supervisión para la cola.

public int QueueStatistics {get;}
Especifica si puede habilitar la recopilación de estadísticas para la cola.

public int QueueType {get;}
El tipo de esta cola con uno de los valores siguientes:

- `MQC.MQQT_ALIAS`
- `MQC.MQQT_LOCAL`
- `MQC.MQQT_REMOTE`
- `MQC.MQQT_CLUSTER`

public int Shareability {get;}
Indica si la cola se puede abrir para entrada varias veces. Los valores posibles son:

- `MQC.MQQA_SHAREABLE`
- `MQC.MQQA_NOT_SHAREABLE`

public string TPIPE {get;}
El nombre de `TPIPE` utilizado para la comunicación con OTMA utilizando el puente IBM MQ IMS .

public int TriggerControl {get; set;}
Indica si los mensajes desencadenantes se graban en una cola de inicio, para iniciar una aplicación para dar servicio a la cola. Los valores posibles son:

- `MQC.MQTC_OFF`
- `MQC.MQTC_ON`

public string TriggerData {get; set;}
Los datos de formato libre que el gestor de colas inserta en el mensaje desencadenante. Inserta `TriggerData` cuando un mensaje que llega a esta cola hace que se grabe un mensaje desencadenante en la cola de inicio. La longitud máxima permitida de la serie la proporciona `MQC.MQ_TRIGGER_DATA_LENGTH`.

public int TriggerDepth {get; set;}
El número de mensajes que deben estar en la cola antes de que se escriba un mensaje desencadenante cuando el tipo de desencadenante se establece en `MQC.MQTT_DEPTH`.

public int TriggerMessagePriority {get; set;}
Prioridad de mensaje bajo la que los mensajes no contribuyen a la generación de mensajes desencadenantes. Es decir, el gestor de colas ignora estos mensajes al decidir si se genera un desencadenante. Un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

public int TriggerType {get; set;}

Las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola. Los valores posibles son:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT_EVERY
- MQC.MQTT_DEPTH

Métodos

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Genera MQException.

Obtiene un mensaje de una cola.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas IBM MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager .

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

Opciones de getMessage

Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions* , la opción de mensaje utilizada es MQGMO_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Tamaño de MaxMsg

El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED .
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED .

Si no se especifica *MaxMsgSize* , se recupera todo el mensaje.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Genera MQException.

Coloca un mensaje en una cola.

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola IBM MQ .

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

mensaje
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

Opciones de putMessage
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

public void PutForwardMessage(MQMessage *message*);
public void PutForwardMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Genera MQException

Coloque un mensaje que se está reenviando en la cola, donde *message* es el mensaje original.

mensaje
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

Opciones de putMessage
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

public void PutReplyMessage(MQMessage *message*)
public void PutReplyMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Coloque un mensaje de respuesta en la cola, donde *message* es el mensaje original.

mensaje
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

Opciones de putMessage
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

public void PutReportMessage(MQMessage *message*)
public void PutReportMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Coloque un mensaje de informe en la cola, donde *message* es el mensaje original.

mensaje
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

Opciones de putMessage
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Constructores

public MQQueue MQQueueManager.AccessQueue(string *queueName*, int *openOptions*);
public MQQueue MQQueueManager.AccessQueue(string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Genera MQException.

Accede a una cola en este gestor de colas.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo `name` del objeto `MQQueue` resultante para averiguar el nombre de la cola dinámica.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF
Utilice el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED
No enlazar a un destino específico.

MQC.MQOO_BIND_ON_OPEN
Manejador de enlace a destino cuando se abre la cola.

MQC.MQOO_BROWSE
Abra para examinar el mensaje.

MQC.MQOO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQOO_INPUT_AS_Q_DEF
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

MQC.MQOO_INPUT_SHARED
Abierto para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE
Abierto para obtener mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE
Abrir para consulta necesaria si desea consultar propiedades.

MQC.MQOO_OUTPUT
Abierto para colocar mensajes.

MQC.MQOO_PASS_ALL_CONTEXT
Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

DynamicQueueName
`dynamicQueueName` se ignora a menos que `queueName` especifique el nombre de una cola modelo. Si lo hace, *`dynamicQueueName`* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si `queueName` especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de

colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

AlternateUserid
Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro `openOptions` , *`alternateUserId`* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , *`alternateUserId`* puede dejarse en blanco o ser nulo.

`public MQQueue(MQQueueManager queueManager, string queueName, int openOptions, string queueManagerName, string dynamicQueueName, string alternateUserId);`

Genera `MQException`.

Accede a una cola en `queueManager`.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo `name` del objeto `MQQueue` resultante para averiguar el nombre de la cola dinámica.

queueManager
Gestor de colas en el que acceder a la cola.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

`MQC.MQOO_ALTERNATE_USER_AUTHORITY`
Validar con el identificador de usuario especificado.

`MQC.MQOO_BIND_AS_QDEF`
Utilice el enlace predeterminado para la cola.

`MQC.MQOO_BIND_NOT_FIXED`
No enlazar a un destino específico.

`MQC.MQOO_BIND_ON_OPEN`
Manejador de enlace a destino cuando se abre la cola.

`MQC.MQOO_BROWSE`
Abra para examinar el mensaje.

`MQC.MQOO_FAIL_IF_QUIESCING`
Fallar si el gestor de colas se está desactivando temporalmente.

`MQC.MQOO_INPUT_AS_Q_DEF`
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

`MQC.MQOO_INPUT_SHARED`
Abierto para obtener mensajes con acceso compartido.

`MQC.MQOO_INPUT_EXCLUSIVE`
Abierto para obtener mensajes con acceso exclusivo.

`MQC.MQOO_INQUIRE`
Abrir para consulta necesaria si desea consultar propiedades.

`MQC.MQOO_OUTPUT`
Abierto para colocar mensajes.

`MQC.MQOO_PASS_ALL_CONTEXT`
Permitir que se pase todo el contexto.

`MQC.MQOO_PASS_IDENTITY_CONTEXT`
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

DynamicQueueName
`dynamicQueueName` se ignora a menos que `queueName` especifique el nombre de una cola modelo. Si lo hace, *`dynamicQueueName`* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si `queueName` especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

AlternateUserid
Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro `openOptions` , *`alternateUserId`* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , *`alternateUserId`* puede dejarse en blanco o ser nulo.

Clase `MQQueueManager`.NET

Utilice `MQQueueManager` para conectarse a un gestor de colas y acceder a los objetos del gestor de colas. También controla las transacciones. El constructor `MQQueueManager` crea una conexión de cliente o servidor.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.ManagedObject
                                │
                                └─ IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```


- “Propiedades” en la página 1809
- “Métodos” en la página 1813
- “Constructores” en la página 1818

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public int AccountingConnOverride {get;}`
Indica si las aplicaciones pueden alterar temporalmente el valor de los valores de contabilidad y de contabilidad de cola de `MQI` .

public int AccountingInterval {get;}
Tiempo antes de que se escriban los registros de contabilidad intermedios (en segundos).

public int ActivityRecording {get;}
Controla la generación de los informes de actividades.

public int AdoptNewMCACheck {get;}
Especifica qué elementos se comprueban para determinar si el MCA se adopta cuando se detecta un nuevo canal de entrada. Para que se adopte, el nombre de MCA debe coincidir con el nombre de un MCA activo.

public int AdoptNewMCAInterval {get;}
Cantidad de tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano.

public int AdoptNewMCAType {get;}
Indica si una instancia de MCA huérfana debe adoptarse (reiniciarse) cuando se detecta una nueva solicitud de canal de entrada que coincide con el valor de MCACheck AdoptNew.

public int BridgeEvent {get;}
Si se generan sucesos de puente IMS .

public int ChannelEvent {get;}
Indica si se generan sucesos de canal.

public int ChannelInitiatorControl {get;}
Si el iniciador de canal se inicia automáticamente cuando se inicia el gestor de colas.

public int ChannelInitiatorAdapters {get;}
Número de subtareas de adaptador para procesar llamadas de IBM MQ .

public int ChannelInitiatorDispatchers {get;}
Número de asignadores a utilizar para el iniciador de canal.

public int ChannelInitiatorTraceAutoStart {get;}
Especifica si el rastreo del iniciador de canal se inicia automáticamente.

public int ChannelInitiatorTraceTableSize {get;}
Tamaño, en megabytes, del espacio de datos de rastreo de un iniciador de canal.

public int ChannelMonitoring {get;}
Si se utiliza la supervisión de canal.

public int ChannelStatistics {get;}
Controla la recopilación de los datos estadísticos para los canales.

public int CharacterSet {get;}
Devuelve el identificador de juego de caracteres codificado (CCSID) del gestor de colas. El gestor de colas utiliza `CharacterSet` para todos los campos de serie de caracteres de la interfaz de programación de aplicaciones.

public int ClusterSenderMonitoring {get;}
Controla la recopilación de datos de supervisión en línea para canales emisores de clúster definidos automáticamente.

public int ClusterSenderStatistics {get;}
Controla la recopilación de datos estadísticos para los canales emisores de clúster definidos automáticamente.

public int ClusterWorkLoadMRU {get;}
Número máximo de canales de clúster de salida.

public int ClusterWorkLoadUseQ {get;}
El valor predeterminado de la propiedad `MQQueue` , `ClusterWorkLoadUseQ`, si especifica un valor de `QMGR`.

public int CommandEvent {get;}
Especifica si se generan sucesos de mandato.

public string CommandInputQueueName {get;}
Devuelve el nombre de la cola de entrada de mandatos definida en el gestor de colas. Las aplicaciones pueden enviar mandatos a esta cola, si están autorizadas a hacerlo.

public int CommandLevel {get;}
Indica el nivel de función del gestor de colas. El conjunto de funciones que corresponden a un nivel de función determinado depende de la plataforma. En una plataforma determinada, puede confiar en que cada gestor de colas dé soporte a las funciones en el nivel funcional más bajo común a todos los gestores de colas.

public int CommandLevel {get;}
Indica si el servidor de mandatos se inicia automáticamente cuando se inicia el gestor de colas.

public string DNSGroup {get;}
Ya no se utiliza.

public int DNSWLM {get;}
Ya no se utiliza.

public int IPAddressVersion {get;}
Qué protocolo IP (IPv4 o IPv6) utilizar para una conexión de canal.

public boolean IsConnected {get;}
Devuelve el valor de `isConnected`.

Si es true, se ha realizado una conexión con el gestor de colas y no se sabe que se ha interrumpido. Las llamadas a IsConnected no intentan activamente acceder al gestor de colas, por lo que es posible que se interrumpa la conectividad física, pero IsConnected todavía puede devolver true. El estado IsConnected sólo se actualiza cuando la actividad, por ejemplo, colocar un mensaje, obtener un mensaje, se realiza en el gestor de colas.

Si es false, no se ha realizado una conexión con el gestor de colas, se ha interrumpido o se ha desconectado.

public int KeepAlive {get;}
Especifica si se va a utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal se cierra.

public int ListenerTimer {get;}
El intervalo de tiempo, en segundos, entre los intentos de IBM MQ de reiniciar el escucha después de una anomalía de APPC o TCP/IP.

public int LoggerEvent {get;}
Indica si se generan sucesos de registrador.

public string LU62ARMSuffix {get;}
El sufijo del miembro APPCPM de SYS1.PARMLIB. Este sufijo designa el LUADD de este iniciador de canal. Cuando el gestor de reinicio automático (ARM) reinicia el iniciador de canal, se emite el mandato z/OS SET APPC=xx.

public string LUGroupName {get; z/os}
Nombre de LU genérico que utilizará el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas.

public string LUName {get;}
El nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 .

public int MaximumActiveChannels {get;}
Número máximo de canales que pueden estar activos en cualquier momento.

public int MaximumCurrentChannels {get;}
El número máximo de canales que pueden ser actuales en cualquier momento (incluidos los canales de conexión de servidor con clientes conectados).

public int MaximumLU62Channels {get;}
El número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 .

public int MaximumMessageLength {get;}
Devuelve la longitud máxima de un mensaje (en bytes) que puede manejar el gestor de colas. No se puede definir ninguna cola con una longitud máxima de mensaje mayor que `MaximumMessageLength`.

public int MaximumPriority {get;}
Devuelve la prioridad máxima de mensajes soportada por el gestor de colas. Las prioridades van de cero (el más bajo) a este valor. Emite `MQException` si llama a este método después de desconectarse del gestor de colas.

public int MaximumTCPChannels {get;}
Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP.

public int MQIAccounting {get;}
Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI).

public int MQIStatistics {get;}
Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas.

public int OutboundPortMax {get;}
El valor máximo en el rango de números de puerto que se van a utilizar al enlazar canales de salida.

public int OutboundPortMin {get;}
El valor mínimo en el rango de números de puerto que se van a utilizar al enlazar canales de salida.

public int QueueAccounting {get;}
Indica si los datos de contabilidad de clase 3 (de nivel de hebra y de nivel de cola) se van a utilizar para todas las colas.

public int QueueMonitoring {get;}
Controla la recopilación de los datos de supervisión para las colas.

public int QueueStatistics {get;}
Controla la recopilación de los datos estadísticos para las colas.

public int ReceiveTimeout {get;}
El periodo de tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado antes de volver al estado inactivo.

public int ReceiveTimeoutMin {get;}
El periodo mínimo de tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado antes de volver a un estado inactivo.

public int ReceiveTimeoutType {get;}
El calificador que se aplica al valor en `ReceiveTimeout`.

public int SharedQueueQueueManagerName {get;}
Especifica cómo entregar mensajes a una cola compartida. Si la transferencia especifica un gestor de colas diferente del mismo grupo de compartición de colas que el gestor de colas de destino, el mensaje se entrega de dos maneras:

MQC.MQSQQM_USE
Los mensajes se entregan al gestor de colas de objetos antes de colocarlos en la cola compartida.

MQCMQSQQM_IGNORE
Los mensajes se colocan directamente en la cola compartida.

public int SSLEvent {get;}
Si se generan sucesos TLS.

public int SSLFips {get;}
Si sólo se van a utilizar algoritmos certificados por FIPS si la criptografía se realiza en IBM MQ, en lugar de en hardware criptográfico.

public int SSLKeyResetCount {get;}
Indica el número de bytes no cifrados enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta.

public int ClusterSenderStatistics {get;}
Especifica el intervalo, en minutos, entre recopilaciones consecutivas de estadísticas.

public int SyncpointAvailability {get;}
Indica si el gestor de colas da soporte a unidades de trabajo y puntos de sincronización con los métodos `MQQueue.get` y `MQQueue.put` .

public string TCPName {get;}
El nombre del único sistema TCP/IP, o el predeterminado, que se va a utilizar, en función del valor de TCPStackType.

public int TCPStackType {get;}
Especifica si el iniciador de canal sólo utiliza el espacio de direcciones TCP/IP especificado en `TCPName`. De forma alternativa, el iniciador de canal puede enlazar con cualquier dirección TCP/IP.

public int TraceRouteRecording {get;}
Controla el registro de la información de rastreo de ruta.

Métodos

public MQProcess AccessProcess(string *processName*, int *openOptions*);
public MQProcess AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera `MQException`.

Acceda a un proceso de IBM MQ en este gestor de colas para consultar los atributos de proceso.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- `MQC.MQOO_FAIL_IF_QUIESCING`
- `MQC.MQOO_INQUIRE`
- `MQC.MQOO_SET`
- `MQC.MQOO_ALTERNATE_USER_AUTHORITY`

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

AlternateUserid

Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro **openOptions** , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica `MQOO_ALTERNATE_USER_AUTHORITY` , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` .

public MQQueue AccessQueue(string *queueName*, int *openOptions*);
public MQQueue AccessQueue(string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Genera `MQException`.

Accede a una cola en este gestor de colas.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo name del objeto `MQQueue` resultante para averiguar el nombre de la cola dinámica.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF
Utilice el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED
No enlazar a un destino específico.

MQC.MQOO_BIND_ON_OPEN
Manejador de enlace a destino cuando se abre la cola.

MQC.MQOO_BROWSE
Abra para examinar el mensaje.

MQC.MQOO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQOO_INPUT_AS_Q_DEF
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

MQC.MQOO_INPUT_SHARED
Abierto para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE
Abierto para obtener mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE
Abrir para consulta necesaria si desea consultar propiedades.

MQC.MQOO_OUTPUT
Abierto para colocar mensajes.

MQC.MQOO_PASS_ALL_CONTEXT
Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

DynamicQueueName
`dynamicQueueName` se ignora a menos que `queueName` especifique el nombre de una cola modelo. Si lo hace, *`dynamicQueueName`* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si `queueName` especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

AlternateUserid
Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro `openOptions` , *`alternateUserId`* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica

MQC.MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede dejarse en blanco o ser nulo.

public MQTopic AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*);
public MQTopic AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*);
public MQTopic AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);
public MQTopic AccessTopic(string *topicName*, string *topicObject*, int *openAs*, int *options*);
public MQTopic AccessTopic(string *topicName*, string *topicObject*, int *openAs*, int *options*, string *alternateUserId*);
public MQTopic AccessTopic(string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic AccessTopic(string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);

Acceda a un tema en este gestor de colas.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, topicObject apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con topicName para crear un nombre de tema. Uno o ambos topicObject o topicName pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en topicObject.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic.Put para publicar en temas, o los métodos MQTopic.Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destino
destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject
En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir series de tema se definen en Combinación de series de tema.

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

AlternateUserid
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName
subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQAsyncStatus GetAsyncStatus();

Genera MQException

Devuelve un objeto MQAsyncStatus , que representa la actividad asíncrona para la conexión del gestor de colas.

public void Backout();

Genera MQException.

Retroceder cualquier mensaje que se haya leído o escrito en el punto de sincronización desde el último punto de sincronización.

Los mensajes que se han escrito con el conjunto de distintivos MQC.MQPMO_SYNCPOINT se eliminan de las colas. Los mensajes leídos con el distintivo MQC.MQGMO_SYNCPOINT se restablecen en las colas de las que proceden. Si los mensajes son persistentes, se registran los cambios.

Para los clientes reconectables, el código de razón MQRC_NONE se devuelve a un cliente después de que la reconexión sea satisfactoria.

public void Begin();

Genera MQException.

Begin sólo está soportado en la modalidad de enlaces de servidor. Inicia una unidad de trabajo global.

public void Commit();

Genera MQException.

Confirme los mensajes que se hayan leído o grabado en el punto de sincronización desde el último punto de sincronización.

Los mensajes escritos con el conjunto de distintivos MQC.MQPMO_SYNCPOINT se ponen a disposición de otras aplicaciones. Los mensajes recuperados con el conjunto de distintivos MQC.MQGMO_SYNCPOINT se suprimen. Si los mensajes son persistentes, se registran los cambios.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si se pierde la conexión al realizar la llamada de confirmación.
- MQRC_BACKED_OUT si la llamada de confirmación se emite después de la reconexión.

Disconnect();

Genera MQException.

Cierre la conexión con el gestor de colas. Todos los objetos a los que se accede en este gestor de colas ya no son accesibles para esta aplicación. Para volver a acceder a los objetos, cree un objeto MQQueueManager .

Generalmente, se confirma cualquier trabajo realizado como parte de una unidad de trabajo. Sin embargo, si la unidad de trabajo está gestionada por .NET, es posible que la unidad de trabajo se retrotraiga.

public void Put(int *type*, string *destinationName*, MQMessage *message*);
public void Put(int *type*, string *destinationName*, MQMessage *message* MQPutMessageOptions *putMessageOptions*);
public void Put(int *type*, string *destinationName*, string *queueManagerName*, string *topicString*, MQMessage *message*);
public void Put(string *queueName*, MQMessage *message*);
public void Put(string *queueName*, MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
public void Put(string *queueName*, string *queueManagerName*, MQMessage *message*);
public void Put(string *queueName*, string *queueManagerName*, MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
public void Put(string *queueName*, string *queueManagerName*, MQMessage *message*, MQPutMessageOptions *putMessageOptions*, string *alternateUserId*);

Genera MQException.

Coloca un solo mensaje en una cola o tema sin crear primero un objeto MQQueue o MQTopic .

queueName
El nombre de la cola en la que colocar el mensaje.

destinationName
El nombre de un objeto de destino. Es una cola o un tema en función del valor de *type*.

tipo
El tipo de objeto de destino. No debe combinar las opciones.

MQC.MQOT_Q
Cola

MQC.MQOT_TOPIC
Tema

queueManagerName

El nombre del gestor de colas o alias de gestor de colas, en el que se define la cola. Si se especifica el tipo `MQC.MQOT_TOPIC` , este parámetro se ignora.

Si la cola es una cola modelo y el nombre del gestor de colas resuelto no es este gestor de colas, se emite un `MQException` .

topicString

`topicString` se combina con el nombre de tema en el objeto de tema *`destinationName`* .

`topicString` se ignora si *`destinationName`* es una cola.

mensaje

El mensaje a enviar. El mensaje es un objeto de entrada/salida.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- `MQRC_CALL_INTERRUPTED` si la conexión se interrumpe al realizar una llamada Put en un mensaje persistente.
- `MQRC_NONE` si la conexión es satisfactoria al realizar una llamada de colocación en un mensaje no persistente (consulte Recuperación de aplicaciones).

Opciones de putMessage

Opciones que controlan las acciones de la colocación.

Si omite *`putMessageOptions`*, se crea una instancia predeterminada de *`putMessageOptions`* . *`putMessageOptions`* es un objeto de entrada/salida.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE` .

AlternateUserid

Especifica un identificador de usuario alternativo utilizado para comprobar la autorización al colocar el mensaje en una cola.

Puede omitir *`alternateUserId`* si no establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en *`putMessageOptions`*. Si establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY`, también debe establecer *`alternateUserId`*. *`alternateUserId`* no tiene efecto a menos que también establezca `MQC.MQOO_ALTERNATE_USER_AUTHORITY`.

Constructores

```
public MQQueueManager();
public MQQueueManager(string queueManagerName);
public MQQueueManager(string queueManagerName, Int options);
public MQQueueManager(string queueManagerName, Int options, string channel,
string connName);
public MQQueueManager(string queueManagerName, string channel, string
connName);
public MQQueueManager(string queueManagerName, System.Collections.Hashtable
properties);
```

Genera `MQException`.

Crea una conexión con un gestor de colas. Seleccione entre crear una conexión de cliente o una conexión de servidor.

Debe tener autorización para inquire (`inq`) en el gestor de colas para conectar con el gestor de colas. Sin autorización para inquire, el intento de conexión falla.

Se crea una conexión de cliente si se cumple una de las condiciones siguientes:

1. *`channel`* o *`connName`* se especifican en el constructor.

2. *HostName*, *Port*o *Channel* se especifican en *properties*.
3. Se especifican *MQEnvironment.HostName*, *MQEnvironment.Port*o *MQEnvironment.Channel* .

Los valores de las propiedades de conexión se establecen de forma predeterminada en el orden que se muestra. *channel* y *connName* en el constructor tienen prioridad sobre los valores de propiedad en el constructor. Los valores de propiedad de constructor tienen prioridad sobre las propiedades MQEnvironment .

El nombre de host, el nombre de canal y el puerto se definen en la clase MQEnvironment .

queueManagerName
Nombre del gestor de colas o grupo de gestores de colas al que conectarse.

Omita el parámetro, o déjelo nulo, o en blanco para realizar una selección de gestor de colas predeterminada. La conexión del gestor de colas predeterminado en un servidor es con el gestor de colas predeterminado en el servidor. La conexión del gestor de colas predeterminado en una conexión de cliente es con el gestor de colas al que está conectado el escucha.

opciones
Especifique las opciones de conexión de MQCNO . Los valores deben ser aplicables al tipo de conexión que se está realizando. Por ejemplo, si especifica las siguientes propiedades de conexión de servidor para una conexión de cliente, se genera un MQException .

- MQC.MQCNO_FASTPATH_BINDING
- MQC.MQCNO_STANDARD_BINDING

properties

El parámetro de propiedades toma una serie de pares de clave/valor que alteran temporalmente las propiedades establecidas por MQEnvironment ; consulte el ejemplo, “Alterar temporalmente las propiedades de MQEnvironment” en la página 1821. Se pueden alterar temporalmente las propiedades siguientes:

- MQC.CONNECT_OPTIONS_PROPERTY
- MQC.CONNECTION_NAME_PROPERTY
- MQC.ENCRYPTION_POLICY_SUITE_B
- MQC.HOST_NAME_PROPERTY
- MQC.PORT_PROPERTY
- MQC.CHANNEL_PROPERTY
- MQC.SSL_CIPHER_SPEC_PROPERTY
- MQC.SSL_PEER_NAME_PROPERTY
- MQC.SSL_CERT_STORE_PROPERTY
- MQC.SSL_CRYPTO_HARDWARE_PROPERTY
- MQC.SECURITY_EXIT_PROPERTY
- MQC.SECURITY_USERDATA_PROPERTY
- MQC.SEND_EXIT_PROPERTY
- MQC.SEND_USERDATA_PROPERTY
- MQC.RECEIVE_EXIT_PROPERTY
- MQC.RECEIVE_USERDATA_PROPERTY
- MQC.USER_ID_PROPERTY
- MQC.PASSWORD_PROPERTY
- MQC.MQAIR_ARRAY
- MQC.KEY_RESET_COUNT
- MQC.FIPS_REQUIRED

- `MQC.HDR_CMP_LIST`
- `MQC.MSG_CMP_LIST`
- `MQC.TRANSPORT_PROPERTY`

canal
Nombre de un canal de conexión de servidor

connName
Nombre de conexión con el formato *HostName* (*Puerto*).

Puede proporcionar una lista de *nombres de host* y *puertos* como argumento para el constructor `MQQueueManager (String queueManagerName, Hashtable properties)` utilizando CONNECTION_NAME_PROPERTY.

Por ejemplo:

```
ConnectionName = "fred.mq.com(2344),nick.mq.com(3746),tom.mq.com(4288)";
Hashtable Properties-new Hashtable();
properties.Add(MQC.CONNECTION_NAME_PROPERTY,ConnectionName);
MQQueueManager qmgr=new MQQueue Manager("qmgrname",properties);
```

Cuando se realiza un intento de conexión, la lista de nombres de conexión se procesa en orden. Si el intento de conexión con el primer nombre de host y puerto falla, se intenta la conexión con el segundo par de atributos. El cliente repite este proceso hasta que se establece una conexión satisfactoria o se agota la lista. Si la lista está agotada, se devuelve un código de razón y un código de terminación adecuados a la aplicación cliente.

Cuando no se proporciona un número de puerto para el nombre de conexión, el puerto predeterminado (configurado en `mqclient.ini`) se utiliza.

Establecer la lista de conexiones

Puede establecer la lista de conexiones utilizando los métodos siguientes cuando se establecen las opciones de reconexión automática de cliente:

Establecer la lista de conexiones mediante MQSERVER
Puede establecer la lista de conexiones a través del indicador de mandatos.

En el indicador de mandatos, establezca el mandato siguiente:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
```

Por ejemplo:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Si establece la conexión en MQSERVER, no la establezca en la aplicación.

Si establece la lista de conexiones en la aplicación, la aplicación sobrescribe cualquier valor establecido en la variable de entorno MQSERVER.

Establecer la lista de conexiones a través de la aplicación
Puede establecer la lista de conexiones en la aplicación especificando el nombre de host y las propiedades de puerto.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Establezca la lista de conexiones a través de `app.config`
`App.config` es un archivo XML donde se especifican los pares de clave-valor.

En la lista de conexiones, especifique

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
<app.Settings>
```

Por ejemplo:

```
<app.Settings>
<add key>="Connection1" value="fred.mq.com(2966)"/>
<add key>="Connection2" value="alex.mq.com(6533)"/>
<app.Settings>
```

Puede cambiar directamente la lista de conexiones en el archivo `app.config` .

Establezca la lista de conexiones a través de `MQEnvironment`

Para establecer la lista de conexiones a través de `MQEnvironment`, utilice la propiedad *ConnectionName* .

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),"alex.mq.com(5211);
```

La propiedad *ConnectionName* sobrescribe las propiedades de nombre de host y puerto establecidas en `MQEnvironment`.

Crear una conexión de cliente

El ejemplo siguiente muestra cómo crear una conexión de cliente con un gestor de colas. Puede crear una conexión de cliente estableciendo las variables `MQEnvironment` antes de crear un nuevo objeto `MQQueueManager` .

```
MQEnvironment.Hostname = "fred.mq.com";  // host to connect to
MQEnvironment.Port     = 1414;           // port to connect to
                                         //If not explicitly set,
                                         // defaults to 1414
                                         // (the default IBM MQ port)
MQEnvironment.Channel  = "channel.name"; // the case sensitive
                                         //  name of the
                                         // SVR CONN channel on
                                         // the queue manager
MQQueueManager qMgr    = new MQQueueManager("MYQM");
```

Figura 11. Conexión de cliente

Alterar temporalmente las propiedades de `MQEnvironment`

El ejemplo siguiente muestra cómo crear un gestor de colas con su ID de usuario y contraseña definidos en una tabla hash.

```
Hashtable properties = new Hashtable();

properties.Add( MQC.USER_ID_PROPERTY, "ExampleUserId" );
properties.Add( MQC.PASSWORD_PROPERTY, "ExamplePassword" );

try
{
    MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
}
catch (MQException mqe)
{
    System.Console.WriteLine("Connect failed with " + mqe.Message);
    return((int)mqe.Reason);
}
```

Figura 12. Alteración temporal de las propiedades de `MQEnvironment`

Crear una conexión reconectable

El ejemplo siguiente muestra cómo volver a conectar automáticamente un cliente a un gestor de colas.

```
Hashtable properties = new Hashtable(); // The queue manager name and the
                                        // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT); // Options
                                        // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY,"fred.mq.com(4789),nick.mq.com(4790)"); // The list
                                        // of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

Figura 13. Reconexión automática de un cliente a un gestor de colas

Clase MQSubscription.NET

Utilice MQSubscription para solicitar que las publicaciones retenidas se envíen al suscriptor. MQSubscription es una propiedad de un objeto MQTopic abierto para suscripción.

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQSubscription
```

public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;

- "Propiedades" en la página 1822
- "Métodos" en la página 1822
- "Constructores" en la página 1823

Propiedades

Acceda a las propiedades de suscripción utilizando la clase MQManagedObject ; consulte "Propiedades" en la página 1780.

Métodos

Acceda a los métodos de suscripción Inquire, Set y Get utilizando la clase MQManagedObject ; consulte "Métodos" en la página 1781.

public int RequestPublicationUpdate(int *options*);

Genera MQException.

Solicite una publicación actualizada para el tema actual. Si el gestor de colas tiene publicaciones retenidas para el tema, se envían al suscriptor.

Antes de llamar a RequestPublicationUpdate, abra un tema para la suscripción para obtener un objeto MQSubscription .

Normalmente, abra la suscripción con la opción MQC.MQSO_PUBLICATIONS_ON_REQUEST . Si no hay comodines en la serie de tema, sólo se envía una publicación como resultado de esta llamada. Si la serie de tema contiene comodines, es posible que se envíen muchas publicaciones. El método devuelve el número de publicaciones retenidas que se envían a la cola de suscripción. No hay garantía de que se reciban tantas publicaciones, especialmente si son mensajes no persistentes.

opciones

MQC.MQSRO_FAIL_IF_QUIESCING
El método falla si el gestor de colas está en un estado de inmovilización. En z/OS, para una aplicación CICS o IMS , `MQC.MQSRO_FAIL_IF_QUIESCING` también fuerza que el método falle si la conexión está en un estado inactivo.

MQC.MQSRO_NONE
No se especifica ninguna opción.

Constructores

No hay ningún constructor público .

Se devuelve un objeto `MQSubscription` en la propiedad `SubscriptionReference` de un objeto `MQTopic` que está abierto para suscripción.

Llame al método `RequestPublicationUpdate` . `MQSubscription` es una subclase de `MQManagedObject`. Utilice la referencia para acceder a las propiedades y métodos de `MQManagedObject`.

Clase MQTopic.NET

Utilice `MQTopic` para publicar o suscribir mensajes sobre un tema, o para consultar o establecer atributos de un tema. Cree un objeto `MQTopic` para publicar o suscribirse utilizando un constructor o el método `MQQueueManager.AccessTopic` .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQDestination
                                        │
                                        └─ IBM.WMQ.MQTopic
```

`public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;`

- “Propiedades” en la página 1823
- “Métodos” en la página 1824
- “Constructores” en la página 1826

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public Boolean IsDurable {get;}
Propiedad de sólo lectura que devuelve `True` si la suscripción es duradera o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá `False`.

public Boolean IsManaged {get;};
Propiedad de sólo lectura que devuelve `True` si la suscripción está gestionada por el gestor de colas, o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá False.

public Boolean IsSubscribed {get;};
Propiedad de sólo lectura que devuelve `True` si el tema se ha abierto para suscripción y `False` si el tema se ha abierto para publicación.

public MQSubscription SubscriptionReference {get;};
Propiedad de sólo lectura que devuelve el objeto MQSubscription asociado con un objeto de tema abierto para suscripción. La referencia está disponible si desea modificar las opciones de cierre o iniciar cualquiera de los métodos de objetos.

public MQDestination UnmanagedDestinationReference {get;};
Propiedad de sólo lectura que devuelve el MQQueue asociado a una suscripción no gestionada. Es el destino especificado cuando se creó el objeto de tema. La propiedad devuelve un valor nulo para los objetos de tema abiertos para publicación o con una suscripción gestionada.

Métodos

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
Emite MQException.

Publica un mensaje en el tema.

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola IBM MQ .

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

mensaje
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

Opciones de putMessage
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);
Emite MQException.

Recupera un mensaje del tema.

Este método utiliza una instancia predeterminada de MQGetMessageOptions para realizar la obtención. La opción de mensaje utilizada es MQGMO_NOWAIT.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas IBM MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager .

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

Opciones de getMessage

Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions* , la opción de mensaje utilizada es MQGMO_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Tamaño de MaxMsg

El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED .
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED .

Si no se especifica *MaxMsgSize* , se recupera todo el mensaje.

Constructores

public MQTopic(MQQueueManager *queueManager*, MQDestination *destination*, string *topicName*, string *topicObject*, int *options*);
public MQTopic(MQQueueManager *queueManager*, MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*);
public MQTopic(MQQueueManager *queueManager*, MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic(MQQueueManager *queueManager*, MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);
public MQTopic(MQQueueManager *queueManager*, string *topicName*, string *topicObject*, int *openAs*, int *options*);
public MQTopic(MQQueueManager *queueManager*, string *topicName*, string *topicObject*, int *openAs*, int *options*, string *alternateUserId*);
public MQTopic(MQQueueManager *queueManager*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic(MQQueueManager *queueManager*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);

Acceda a un tema en *queueManager*.

Los objetos de `MQTopic` están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, `topicObject` apunta a un objeto de tema administrativo. El constructor `MQTopic` obtiene una serie de tema del objeto de tema y la combina con `topicName` para crear un nombre de tema. Uno o ambos `topicObject` o `topicName` pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en `topicObject`.

Los temas que están asociados con el objeto `MQTopic` son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto `MQTopic` puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos `MQTopic.Put` para publicar en temas, o los métodos `MQTopic.Get` para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto `MQTopic` para la suscripción, sin proporcionar un objeto `MQDestination` , se presupone una suscripción gestionada. Si pasa una cola como un objeto `MQDestination` , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

queueManager
Gestor de colas sobre el que acceder a un tema.

destino
destination es una instancia de `MQQueue` . Al proporcionar *destination*, `MQTopic` se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject
En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena

con *topicName*. Las reglas para construir series de tema se definen en Combinación de series de tema.

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

AlternateUserid
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName
subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQTopic MQQueueManager.AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*);
public MQTopic MQQueueManager.AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*);
public MQTopic MQQueueManager.AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic MQQueueManager.AccessTopic(MQDestination *destination*, string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);
public MQTopic MQQueueManager.AccessTopic(string *topicName*, string *topicObject*, int *openAs*, int *options*);
public MQTopic MQQueueManager.AccessTopic(string *topicName*, string *topicObject*, int *openAs*, int *options*, string *alternateUserId*);
public MQTopic MQQueueManager.AccessTopic(string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*);
public MQTopic MQQueueManager.AccessTopic(string *topicName*, string *topicObject*, int *options*, string *alternateUserId*, string *subscriptionName*, System.Collections.Hashtable *properties*);

Acceda a un tema en este gestor de colas.

Los objetos de `MQTopic` están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, `topicObject` apunta a un objeto de tema administrativo. El constructor `MQTopic` obtiene una serie de tema del objeto de tema y la combina con `topicName` para crear un nombre de tema. Uno o ambos `topicObject` o `topicName` pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en `topicObject`.

Los temas que están asociados con el objeto `MQTopic` son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *`topicObject`*. La segunda serie de tema es *`topicString`*. La serie de tema resultante asociada con el objeto `MQTopic` puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos `MQTopic.Put` para publicar en temas, o los métodos `MQTopic.Get` para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto `MQTopic` para la suscripción, sin proporcionar un objeto `MQDestination` , se presupone una suscripción gestionada. Si pasa una cola como un objeto `MQDestination` , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destino
`destination` es una instancia de `MQQueue` . Al proporcionar *`destination`*, `MQTopic` se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *`destination`*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *`topicName`* se concatena con la serie de tema definida en el objeto de tema administrativo *`topicObject`* . Puede establecer *`topicName`* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *`topicObject`*.

topicObject
En la entrada, *`topicObject`* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *`topicObject`* se concatena con *`topicName`*. Las reglas para construir series de tema se definen en Combinación de series de tema.

En la salida, *`topicObject`* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- `MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION`
- `MQC.MQTOPIC_OPEN_AS_PUBLICATION`

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de `MQC.MQSO_*` para acceder a un tema para la suscripción y las constantes de `MQC.MQOO_*` para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador `OR` a nivel de bit.

AlternateUserid
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *`alternateUserId`*, si `MQC.MQOO_ALTERNATE_USER_AUTHORITY` o `MQC.MQSO_ALTERNATE_USER_AUTHORITY` se ha establecido en el parámetro de opciones.

subscriptionName
`subscriptionName` es necesario si se proporcionan las opciones `MQC.MQSO_DURABLE` o `MQC.MQSO_ALTER` . En ambos casos, `MQTopic` se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido `MQC.MQSO_DURABLE` y la suscripción existe, o si se ha establecido `MQC.MQSO_ALTER` y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- `MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID`
- `MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY`
- `MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA`
- `MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID`
- `MQC.MQSUB_PROP_PUBLICATION_PRIORITY`
- `MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN`
- `MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA`

Interfaz de `IMQObjectTrigger`.NET

Implemente `IMQObjectTrigger` para procesar los mensajes pasados por el supervisor de **runmqdnm**.NET .

Interface

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

En función de si se especifica el control de punto de sincronización en el mandato **runmqdnm** , el mensaje se elimina de la cola antes o después de que se devuelva el método `Execute` .

Métodos

void Execute (MQQueueManager *queueManager*, MQQueue *queue*, MQMessage *message*, string *param*);

queueManager
Gestor de colas que aloja la cola que se está supervisando.

cola
Cola que se está supervisando.

mensaje
Mensaje leído de la cola.

param
Datos pasados desde `UserParameter`.

Interfaz de MQC.NET

Consulte una constante `MQI` añadiendo el prefijo `MQC.` al nombre de la constante. `MQC` define todas las constantes utilizadas por `MQI`.

Interface

```
System.Object
        |
        └ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Ejemplo

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

Identificadores de juego de caracteres para aplicaciones .NET

Descripciones de los juegos de caracteres que puede seleccionar para codificar mensajes de .NET IBM MQ

Juego de caracteres	Descripción
37	ibm037
437	ibm437 /PC Original
500	ibm500
819	iso-8859-1 / latin1 / ibm819
1200	Unicode
1208	UTF-8
273	ibm273
277	ibm277
278	ibm278
280	ibm280
284	ibm284

Juego de caracteres	Descripción
285	ibm285
297	ibm297
420	ibm420
424	ibm424
737	ibm737 /PC Griego
775	ibm775 /PC Báltico
813	iso-8859-7 /greek/ ibm813
838	ibm838
850	ibm850 /PC Latin 1
852	ibm852 /PC Latin 2
855	ibm855 /PC Cirílico
856	ibm856
857	ibm857 /PC Turco
860	ibm860 /PC Portugués
861	ibm861 /PC islandés
862	ibm862 /PC Hebreo
863	ibm863 /PC francés canadiense
864	ibm864 /PC árabe
865	ibm865 /PC Nordic
866	ibm866 /PC Ruso
868	ibm868
869	ibm869 /PC Modern Griego
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 /cirílico/ ibm915
916	iso-8859-8 /hebreo/ ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922

Juego de caracteres	Descripción
930	ibm930
932	PC japonés
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 /Big 5 Chino tradicional
954	eucjis
964	ibm964 /CNS 11643 Chino tradicional
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 /arabic/ ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latín 2
1251	Windows Cirílico
1252	Windows Latín 1
1253	Windows Griego
1254	Windows Turco
1255	Windows Hebreo
1256	Windows Árabe
1257	Windows Bático
1258	Windows Vietnamita
1381	ibm1381
1383	ibm1383

Juego de caracteres	Descripción
2022	JIS
5601	ksc-5601 Coreano
33722	ibm33722

Clases C++ de IBM MQ

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

Para cada clase, se muestra la siguiente información:

Diagrama de jerarquía de clases
Un diagrama de clase que muestra la clase en su relación de herencia con sus clases padre inmediatas, si las hay.

Otras clases pertinentes
El documento enlaza con otras clases relevantes, como las clases padre, y las clases de objetos utilizados en las firmas de método.

Atributos de objetos
Atributos de la clase. Estos atributos se añaden a los definidos para cualquier clase padre. Muchos atributos reflejan miembros de estructura de datos de IBM MQ (consulte "Referencia cruzada de C++ y MQI" en la página 1834). Para obtener descripciones detalladas, consulte "objeto, atributos de" en la página 810.

Constructores
Firmas de los métodos especiales utilizados para crear un objeto de la clase.

Métodos de objeto (public)
Firmas de métodos que requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Cuando sea aplicable, también se mostrará la siguiente información:

Métodos de clase (public)
Firmas de métodos que no requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Métodos sobrecargados (clase padre)
Firmas de los métodos virtuales definidos en las clases padre, pero que muestran un comportamiento polimórfico diferente para esta clase.

Métodos de objeto (protegidos)
Firmas de métodos que requieren una instancia de la clase para su operación y que están reservadas para que las utilicen las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

Datos de objeto (protegidos)
Detalles de implementación para datos de instancia de objeto disponibles para las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

códigos de razón
Valores MQRC_ * (consulte Códigos de terminación y razón de la API) que se puede esperar de los métodos que fallan. Para obtener una lista exhaustiva de los códigos de razón que se pueden producir para un objeto de una clase, consulte la documentación de la clase padre. La lista documentada de códigos de razón para una clase no incluye los códigos de razón para las clases padre.

Nota:

1. Los objetos de estas clases no son seguros para hebras. Esto garantiza un rendimiento óptimo, pero tenga cuidado de no acceder a ningún objeto desde más de una hebra.

2. Se recomienda que, para un programa multihebra, se utilice un objeto de gestor ImqQueueindependiente para cada hebra. Cada objeto de gestor debe tener su propia colección independiente de otros objetos, asegurándose de que los objetos de distintas hebras estén aislados entre sí.

Las clases son:

- "Clase C++ de registro ImqAuthentication" en la página 1852
- "Clase C++ ImqBinary" en la página 1854
- "Clase C++ ImqCache" en la página 1856
- "Clase C++ ImqChannel" en la página 1859
- "ImqCICSBridgeHeader clase C++" en la página 1864
- "Clase C++ ImqDeadLetterHeader" en la página 1870
- "ImqDistributionListar clase C++" en la página 1873
- "Clase C++ ImqError" en la página 1874
- "Clase C++ ImqGetMessageOptions" en la página 1875
- "Clase C++ ImqHeader" en la página 1879
- "ImqIMSBridgeHeader clase C++" en la página 1880
- "Clase C++ ImqItem" en la página 1883
- "Clase C++ ImqMessage" en la página 1885
- "Clase C++ de rastreador ImqMessage" en la página 1892
- "Clase C++ ImqNamelist" en la página 1895
- "Clase C++ ImqObject" en la página 1896
- "Clase C++ ImqProcess" en la página 1902
- "Clase C++ ImqPutMessageOptions" en la página 1904
- "Clase C++ ImqQueue" en la página 1906
- "Clase C++ del gestor de ImqQueue" en la página 1917
- "Clase C++ de cabecera ImqReference" en la página 1933
- "Clase C++ ImqString" en la página 1936
- "Clase C++ ImqTrigger" en la página 1941
- "Clase C++ de cabecera ImqWork" en la página 1944

Referencia cruzada de C++ y MQI

Esta colección de temas contiene información relacionada con C++ para la MQI.

Lea esta información junto con "Tipos de datos utilizados en la MQI" en la página 235.

Esta tabla relaciona estructuras de datos MQI con las clases C++ y archivos de inclusión. Los temas siguientes muestran información de referencia cruzada para cada clase C++. Estas referencias cruzadas están relacionadas con el uso de las interfaces de procedimiento de IBM MQ subyacentes. Las clases ImqBinary, ImqDistributionList e ImqString no tienen atributos que entren en esta categoría y se excluyen.

Tabla 845. Estructura de datos, clase y referencia cruzada de archivo de inclusión

Estructura de datos	Clase	Archivo Include
MQAIRE	Registro ImqAuthentication	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp

Tabla 845. Estructura de datos, clase y referencia cruzada de archivo de inclusión (continuación)		
Estructura de datos	**Clase**	**Archivo Include**
MQCD	ImqChannel	imqchl.hpp
MQCIH	ImqCICSBridgeHeader	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	Lista de ImqDistribution	imqdst.hpp
	ImqError	imqerr.hpp
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	ImqIMSBridgeHeader	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	Rastreador de ImqMessage	imqmtr.hpp
	ImqNamelist	imqnml.hpp
MQOD, MQRR	ImqObject	imqobj.hpp
MQPMO, MQPMR, MQRR	ImqPutMessageOptions	imqpmo.hpp
	ImqProcess	imqpro.hpp
	ImqQueue	imqque.hpp
MQBO, MQCNO, MQCSP	Gestor de ImqQueue	imqmgr.hpp
MQRMH	Cabecera ImqReference	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	Cabecera ImqWork	imqwih.hpp

Referencia cruzada de registro ImqAuthentication

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ de registro ImqAuthentication.

Tabla 846. Atributos, estructuras de datos, campos y llamadas			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
Nombre de conexión	MQAIRE	AuthInfoConnName	MQCONNX
contraseña	MQAIRE	LDAPPassword	MQCONNX
tipo	MQAIRE	AuthInfoType	MQCONNX
nombre de usuario	MQAIRE	LDAPUserNamePtr	MQCONNX

Tabla 846. Atributos, estructuras de datos, campos y llamadas (continuación)			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
	MQAIRE	Desplazamiento de LDAPUserName	MQCONNX
	MQAIRE	LDAPUserNameLongitud	MQCONNX

Referencia cruzada de ImqCache

Referencia cruzada de atributos y llamadas para la clase C++ ImqCache .

Tabla 847. Atributos y llamadas	
Atributo	**Llamada**
almacenamiento intermedio automático	MQGET
Longitud de almacenamiento intermedio	MQGET
puntero de almacenamiento intermedio	MQGET, MQPUT
Longitud de datos	MQGET
Desplazamiento de datos	MQGET
puntero de datos	MQGET
Longitud del mensaje	MQGET, MQPUT

Referencia cruzada de ImqChannel

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqChannel .

Tabla 848. Atributos, estructuras de datos, campos y llamadas			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
latido del corazón por lotes	MQCD	BatchHeartbeat	MQCONNX
nombre de canal	MQCD	ChannelName	MQCONNX
Nombre de conexión	MQCD	ConnectionName	MQCONNX
	MQCD	Nombre de ShortConnection	MQCONNX
Compresión de cabecera	MQCD	Lista de HdrComp	MQCONNX
intervalo de latido cardíaco	MQCD	HeartbeatInterval	MQCONNX
Intervalo de mantenimiento activado	MQCD	Intervalo de mantenimiento de activación	MQCONNX
dirección local	MQCD	LocalAddress	MQCONNX
longitud máxima del mensaje	MQCD	MaxMsgLength	MQCONNX
Compresión de mensaje	MQCD	Lista de MsgComp	MQCONNX
nombre modalidad	MQCD	ModeName	MQCONNX
contraseña	MQCD	Contraseña	MQCONNX
recuento de salidas de recepción	MQCD		MQCONNX

Tabla 848. Atributos, estructuras de datos, campos y llamadas (continuación)			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
nombres de salida de recepción	MQCD	ReceiveExit	MQCONNX
	MQCD	ReceiveExitsdefinido	MQCONNX
	MQCD	ReceiveExitPtr	MQCONNX
recibir datos de usuario	MQCD	ReceiveUserData	MQCONNX
	MQCD	ReceiveUserDataPtr	MQCONNX
Nombre de salida de seguridad	MQCD	SecurityExit	MQCONNX
datos de usuario de seguridad	MQCD	Datos de SecurityUser	MQCONNX
recuento de salidas de envío	MQCD		MQCONNX
nombres de salida de envío	MQCD	SendExit	MQCONNX
	MQCD	SendExitsdefinido	MQCONNX
	MQCD	SendExitPtr	MQCONNX
enviar datos de usuario	MQCD	SendUserData	MQCONNX
	MQCD	SendUserDataPtr	MQCONNX
CipherSpec de SSL	MQCD	Especificación sslCipher	MQCONNX
Tipo de autenticación de cliente SSL	MQCD	Autenticación de sslClient	MQCONNX
Nombre de igual SSL	MQCD	sslPeerName	MQCONNX
nombre del programa de transacciones	MQCD	TpName	MQCONNX
tipo de transporte	MQCD	TransportType	MQCONNX
ID de usuario	MQCD	UserIdentifier	MQCONNX

Referencia cruzada deCICSBridgeHeader de Imq

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++CICSBridgeHeader de Imq.

Tabla 849. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
código de terminación anómala de puente	MQCIH	AbendCode
Descriptor ADS	MQCIH	AdsDescriptor
identificador de atención	MQCIH	AttentionId
autenticador	MQCIH	Authenticator
código de terminación de puente	MQCIH	Código de BridgeCompletion
desplazamiento de error de puente	MQCIH	ErrorOffset
código de razón de puente	MQCIH	BridgeReason
código de cancelación de puente	MQCIH	CancelCode
tarea conversacional	MQCIH	ConversationalTask

Tabla 849. Correlación de atributos, estructuras de datos y campos (continuación)		
Atributo	**Estructura de datos**	**Campo**
Posición del cursor	MQCIH	CursorPosition
señal de recurso	MQCIH	Facility
tiempo de mantenimiento del recurso	MQCIH	FacilityKeepTime
recurso como	MQCIH	FacilityLike
función	MQCIH	Función
obtener intervalo de espera	MQCIH	GetWaitInterval
Tipo de enlace	MQCIH	LinkType
identificador de transacción siguiente	MQCIH	NextTransactionId
longitud de datos de salida	MQCIH	OutputDataLength
formato de respuesta	MQCIH	ReplyToFormat
código de retorno de puente	MQCIH	ReturnCode
código de inicio	MQCIH	StartCode
estado de finalización de tarea	MQCIH	TaskEndStatus
Identificador de transacción	MQCIH	TransactionId
control uow	MQCIH	UowControl
versión	MQCIH	Versión

ImqDeadLetterHeader referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqDeadLetterHeader .

Tabla 850. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
código de razón de mensaje no enviado	MQDLH	Razón
Nombre del gestor de colas de destino	MQDLH	DestQMgrName
Nombre de la cola de destino	MQDLH	DestQName
Nombre de aplicación de transferencia	MQDLH	PutApplName
Tipo de aplicación de transferencia	MQDLH	PutApplType
Fecha de transferencia	MQDLH	PutDate
Hora de transferencia	MQDLH	PutTime

Referencia cruzada ImqError

Referencia cruzada de atributos y llamadas para la clase C++ ImqError .

Tabla 851. Atributos y llamadas	
Atributo	**Llamada**
código de terminación	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

Tabla 851. Atributos y llamadas (continuación)	
Atributo	**Llamada**
código de razón	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

ImqGetMessageOptions referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqGetMessageOptions .

Tabla 852. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
Estado de grupo	MQGMO	GroupStatus
opciones de coincidencia	MQGMO	MatchOptions
token de mensaje	MQGMO	MessageToken
opciones	MQGMO	Opciones
Nombre de cola resuelto	MQGMO	ResolvedQName
longitud devuelta	MQGMO	ReturnedLength
segmentación	MQGMO	Segmentation
estado de segmento	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
participación de punto de sincronismo	MQGMO	Opciones
Intervalo de espera	MQGMO	WaitInterval

Referencia cruzada de ImqHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqHeader .

Tabla 853. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
juego de caracteres	MQDLH, MQIIH	CodedCharSetId
codificación	MQDLH, MQIIH	Codificación
formato	MQDLH, MQIIH	Formato
distintivos de cabecera	MQIIH, MQRMH	Distintivos

Referencia cruzada deIMSBridgeHeader de Imq

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 854. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
autenticador	MQIIH	Authenticator

Tabla 854. Correlación de atributos, estructuras de datos y campos (continuación)		
Atributo	**Estructura de datos**	**Campo**
Modalidad de confirmación	MQIIH	CommitMode
Alteración temporal del terminal lógico	MQIIH	LTermOverride
Nombre de la correlación de servicios del formato del mensaje	MQIIH	MFSMapName
formato de respuesta	MQIIH	ReplyToFormat
Ámbito de seguridad	MQIIH	SecurityScope
id de instancia de transacción	MQIIH	TranInstanceId
ESTADO DE TRANSACCIÓN	MQIIH	TranState

Referencia cruzada de ImqItem

Referencia cruzada de atributos y llamadas para la clase C++ ImqItem .

Tabla 855. Atributos y llamadas	
Atributo	**Llamada**
id de estructura	MQGET

Referencia cruzada de ImqMessage

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqMessage .

Tabla 856. Atributos, estructuras de datos, campos y llamadas			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
Datos de ID de aplicación	MQMD	ApplIdentityData	
Datos de origen de aplicación	MQMD	ApplOriginData	
Recuento de restituciones	MQMD	BackoutCount	
juego de caracteres	MQMD	CodedCharSetId	
codificación	MQMD	Codificación	
caducidad	MQMD	Caducidad	
formato	MQMD	Formato	
Distintivos de mensaje	MQMD	MsgFlags	
tipo de mensaje	MQMD	MsgType	
offset	MQMD	Desplazamiento	
Longitud original	MQMD	OriginalLength	
persistencia	MQMD	Persistence	
priority	MQMD	Priority	
Nombre de aplicación de transferencia	MQMD	PutApplName	
Tipo de aplicación de transferencia	MQMD	PutApplType	

Tabla 856. Atributos, estructuras de datos, campos y llamadas (continuación)			
Atributo	**Estructura de datos**	**Campo**	**Llamada**
Fecha de transferencia	MQMD	PutDate	
Hora de transferencia	MQMD	PutTime	
Nombre de gestor de cola de respuestas	MQMD	GestorColasRespuesta	
Nombre de cola de respuestas	MQMD	ReplyToQ	
informe	MQMD	Informe	
número de secuencia	MQMD	MsgSeqNumber	
longitud total de mensaje		DataLength	MQGET
ID de usuario	MQMD	UserIdentifier	

Referencia cruzada de ImqMessageTracker

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de rastreador ImqMessage.

Tabla 857. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
Señal de contabilidad	MQMD	AccountingToken
ID de correlación	MQMD	CorrelId
información de retorno	MQMD	Comentarios
ID de grupo	MQMD	GroupId
ID de mensaje	MQMD	MsgId

Referencia cruzada de ImqNamelist

Referencia cruzada de atributos, consultas y llamadas para la clase C++ ImqNamelist .

Tabla 858. Atributos, consultas y llamadas		
Atributo	**Consulta**	**Llamada**
Número de nombres	MQIA_NAME_COUNT	MQINQ
Nombre de lista de nombres	MQCA_NAMELIST_NAME	MQINQ

Referencia cruzada de ImqObject

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqObject .

Tabla 859. Atributos, estructuras de datos, campos, consultas y llamadas				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Fecha de modificación			MQCA_ALTERATION_DATE	MQINQ
Hora de modificación			MQCA_ALTERATION_TIME	MQINQ

Tabla 859. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
ID de usuario alternativo	MQOD	AlternateUserId		
ID de seguridad alternativo				
opciones de cierre				MQCLOSE
description			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nombre	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opciones abiertas				MQOPEN
estado abierto				MQOPEN, MQCLOSE
identificador de gestor de colas	identificador de gestor de colas		MQCA_Q_MGR_IDENTIFIER	MQINQ

Referencia cruzada de ImqProcess

Referencia cruzada de atributos, consultas y llamadas para la clase C++ de registro ImqAuthentication.

Tabla 860. Atributos, consultas y llamadas		
Atributo	**Consulta**	**Llamada**
ID de aplicación	MQCA_APPL_ID	MQINQ
Tipo de aplicación	MQIA_APPL_TYPE	MQINQ
Datos de entorno	MQCA_ENV_DATOS	MQINQ
datos de usuario	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 861. Correlación de atributos, estructuras de datos y campos		
Atributo	**Estructura de datos**	**Campo**
referencia de contexto	MQPMO	Contexto
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
opciones	MQPMO	Opciones
campos de registro	MQPMO	PutMsgRecFields

Tabla 861. Correlación de atributos, estructuras de datos y campos (continuación)		
Atributo	**Estructura de datos**	**Campo**
Nombre del gestor de colas resuelto	MQPMO	ResolvedQMgrName
Nombre de cola resuelto	MQPMO	ResolvedQName
	MQPMO	Tiempo de espera
	MQPMO	UnknownDestCount
participación de punto de sincronismo	MQPMO	Opciones

Referencia cruzada de ImqQueue

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqQueue .

Tabla 862. Referencia cruzada de ImqQueue				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Nombre de reposición en cola para restitución			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
umbral de restituciones			Umbral MQIA_BACKOUT_THRESHOLD	MQINQ
nombre de cola base			MQCA_BASE_Q_NAME	MQINQ
Nombre de clúster			NOMBRE_CLÚSTER MQCA_clúster	MQINQ
Nombre de la lista de nombres del clúster			MQCA_XX_ENCODE_CASE_ONE nombre_clúster	MQINQ
Rango de carga trabajo del clúster			MQIA_CLWL_Q_RANK	MQINQ
Prioridad de carga de trabajo del clúster			MQIA_CLWL_Q_PRIORITY	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
fecha de creación			MQCA_CREATION_DATE	MQINQ
Hora de creación			MQCA_CREATION_TIME	MQINQ
Profundidad actual			MQIA_PROFUNDIDAD_Q_ACTUAL	MQINQ
enlace predeterminado			MQIA_DEF_BIND	MQINQ
Opción abierta de entrada predeterminada			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
Persistencia predeterminada			MQIA_PERSISTENCIA	MQINQ
Prioridad predeterminada			MQIA_PRIORIDAD_DEF_PRIORIDAD	MQINQ

Tabla 862. Referencia cruzada de ImqQueue (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Tipo de definición			MQIA_DEFINITION_TYPE	MQINQ
suceso de profundidad alta			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
límite alto de profundidad			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
suceso de profundidad baja			MQIA_Q_DEPTH_LOW_EVENT	MQINQ
límite bajo de profundidad			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
suceso de profundidad máxima			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
listas de distribución			MQIA_DIST_LISTS	MQINQ, MQSET
Nombre de la cola dinámica	MQOD	DynamicQName		
Copia en disco de restitución de obtención			MQIA_HARDEN_GET_BACKOUT	MQINQ
Tipo de índice			TIPO_ÍNDICE	MQINQ
inhibir obtención			INHIBIDORES DE MQIA_GET	MQINQ, MQSET
inhibir colocación			INHIBIDORES DE MQIA_PUT	MQINQ, MQSET
Nombre cola iniciación			MQCA_INITIATION_Q_NAME	MQINQ
Profundidad máxima			MQIA_MAX_Q_DEPTH	MQINQ
longitud máxima del mensaje			MQIA_MAX_MSG_LENGTH	MQINQ
Secuencia de entrega de mensajes			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
siguiente cola distribuida				
Clase de mensajes no permanentes			MQIA_NPM_CLASS	MQINQ
Cuenta de entradas abiertas			MQIA_RECUENTO_ENTRADA_ABIERTA	MQINQ
Cuenta de salidas abiertas			MQIA_RECUENTO_SALIDA_ABIERTA	MQINQ
cola distribuida anterior				

Tabla 862. Referencia cruzada de ImqQueue (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
nombre de proceso			MQCA_PROCESS_NAME	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
nombre del gestor de colas	MQOD	ObjectQMgrName		
Supervisión de la cola			MQIA_MONITORING_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tipo de cola			MQIA_Q_TYPE	MQINQ
Nombre de gestor de colas remoto			MQCA_REMOTE_Q_MGR_NAME	MQINQ
Nombre de cola remota			MQCA_REMOTE_Q_NAME	MQINQ
Nombre del gestor de colas resuelto	MQOD	ResolvedQMgrName		
Nombre de cola resuelto	MQOD	ResolvedQName		
Intervalo de retención			MQIA_INTERVALO_RETENCIÓN	MQINQ
ámbito			MQIA_ÁMBITO	MQINQ
intervalo de servicio			MQIA_Q_SERVICE_INTERVAL	MQINQ
suceso de intervalo de servicio			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
Posibilidad de compartición			MQIA_COMPARTIBILIDAD	MQINQ
clase de almacenamiento			CLASE_ALMACENAMIENTO_MQCA	MQINQ
Nombre de cola de transmisión			MQCA_XMIT_Q_NAME	MQINQ
Activar control			MQIA_TRIGGER_CONTROL	MQINQ, MQSET
Datos desencadenantes			MQCA_TRIGGER_DATA	MQINQ, MQSET
Profundidad de desencadenante			MQIA_PROFUNDIDAD	MQINQ, MQSET
Prioridad de mensajes desencadenantes			MQIA_TRIGGER_MSG_PRIORITY	MQINQ, MQSET
tipo de activador			MQIA_TIPO_TRIGGER_TYPE	MQINQ, MQSET

Tabla 862. Referencia cruzada de ImqQueue (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
utilización			MQIA_USAGE	MQINQ

Referencia cruzada de ImqQueueManager

Referencias cruzadas de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ del gestor ImqQueue.

Tabla 863. Atributos, estructuras de datos, campos, consultas y llamadas				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
alteración temporal de conexiones de contabilidad			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
Intervalo contable			MQIA_INTERVALO_CONTABILIDAD	MQINQ
registro de actividad			MQIA_GRABACIÓN_ACTIVIDAD	MQINQ
Adoptar nueva comprobación MCA			MQIA_ADOPTNEWMCA_CHECK	MQINQ
Adoptar nuevo tipo MCA			MQIA_ADOPTNEWMCA_TYPE	MQINQ
Tipo de autenticación	MQCSP	AuthenticationType		MQCONNX
suceso de autorización			SUCESO_AUTORIZACIÓN_MQIA_EVENT	MQINQ
Opciones de inicio	MQBO	Opciones		MQBEGIN
suceso de puente			MQIA_SUCESO_PUENTE	MQINQ
Definición automática de canal			MQIA_CANAL_AUTO_DEF	MQINQ
suceso de definición automática de canal			MQIA_SUCESO_AUTO_CANAL	MQIA
Salida de definición automática de canal			MQIA_SALIDA_AUTO_CANAL	MQIA
suceso de canal			MQIA_SUCESO_CANAL	MQINQ
Adaptadores del iniciador de canal			MQIA_CHINIT_ADAPTERS	MQINQ

Tabla 863. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Control del iniciador de canal			CONTROL MQIA_CHINIT_DE	MQINQ
Asignadores de tareas del iniciador de canal			MQIA_CHINIT_DISPATCHERS	MQINQ
Inicio automático del rastreo del iniciador de canal			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Tamaño de tabla de rastreo del iniciador de canal			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
Supervisión de canal			MQIA_MONITORING_CHANNEL	MQINQ
referencia de canal	MQCD	ChannelType		MQCONNX
Estadísticas del canal			MQIA_STATISTICS_CHANNEL	MQINQ
juego de caracteres			MQIA_CODED_CHAR_SET_ID	MQINQ
Supervisión de clúster emisor			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Estadística de clúster emisor			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Datos de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_DATA	MQINQ
salida de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
Longitud de la carga de trabajo de clúster			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
mru de carga de trabajo de clúster			MQIA_CLWL_MRU_CHANNELS	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
suceso de mandato			MQIA_SUCESO_MANDATO	MQINQ
Nombre de cola de entrada de mandatos			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
Nivel de mandato			MQIA_COMMAND_LEVEL	MQINQ

Tabla 863. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Control del servidor de mandatos			MQIA_CMD_SERVER_CONTROL	MQINQ
Opciones de conexión	MQCNO	Opciones		MQCONN, MQCONNX
ID de conexión	MQCNO	ConnectionId		MQCONNX
estado de conexión				MQCONN, MQCONNX, MQDISC
etiqueta de conexión	MQCD	ConnTag		MQCONNX
hardware de cifrado	MQSCO	CryptoHardware		MQCONNX
nombre de cola de mensajes no entregados			MQCA_DEAD_LETTER_Q_NAME	MQINQ
nombre de cola de transmisión predeterminada			MQCA_DEF_XMIT_Q_NAME	MQINQ
listas de distribución			MQIA_DIST_LISTS	MQINQ
grupo dns			MQCA_DNS_GROUP	MQINQ
dns wlm			MQIA_DNS_WLM	MQINQ
primer registro de autenticación	MQSCO	AuthInfoRecOffset		MQCONNX
	MQSCO	AuthInfoRecPtr		MQCONNX
suceso de inhibición			SUCESO INHIBIDOR DE MQIA_ENT	MQINQ
Versión de dirección IP			MQIA_IP_ADDRESS_VERSION	MQINQ
repositorio de claves	MQSCO	KeyRepository		MQCONNX
recuento de restablecimiento de clave	MQSCO	Recuento de KeyReset		MQCONNX
Temporizador de escucha			MQIA_LISTENER_TIMER	MQINQ
suceso local			MQIA_SUCESO_LOCAL	MQINQ
LoggerEvent			MQIA_LOGGER_EVENT	MQINQ

Tabla 863. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Nombre de grupo LU			MQCA_LU_XX_ENCODE_CASE_ONE nombre_grupo	MQINQ
Nombre de LU			MQCA_LU_NAME	MQINQ
Sufijo de brazo lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
Canales lu62			MQIA_LU62_CHANNELS	MQINQ
máximo de canales activos			MQIA_CANALES_ACTIVOS	MQINQ
Canales máximos			MQIA_MÁX_CANALES	MQINQ
máximo de manejadores			MQIA_MÁX_DESCRIPTORES de contexto	MQINQ
longitud máxima del mensaje			MQIA_MAX_MSG_LENGTH	MQINQ
Prioridad máxima			MQIA_PRIORIDAD_MÁX	MQINQ
Máx. mensajes no confirmados			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
Contabilidad de MQI			MQIA_CUENTA_MQI	MQINQ
Estadísticas MQI			MQIA_ESTADÍSTICAS_MQI	MQINQ
máximo de puerto de salida			MQIA_PUERTO_SALIDA máx	MQINQ
mínimo de puerto de salida			MQIA_PUERTO_SALIDA min	MQINQ
contraseña	MQCSP	CSPPasswordPtr		MQCONNX
	MQCSP	CSPPasswordOffset		MQCONNX
	MQCSP	CSPPasswordLength		MQCONNX
suceso de rendimiento			MQIA_SUCESO_RENDIMIENTO	MQINQ
plataforma			MQIA_PLATFORM	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
Supervisión de la cola			MQIA_MONITORING_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tiempo de espera de recepción			MQIA_TIEMPO_ESPERA	MQINQ

Tabla 863. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
tiempo de espera de recepción mínimo			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Tipo de tiempo de espera de recepción			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
suceso remoto			MQIA_REMOTE_EVENT	MQINQ
Nombre de depósito			MQCA_REPOSITORY_NAME	MQINQ
Lista nombres repositorio			MQCA_REPOSITORY_NAMELIST	MQINQ
nombre de gestor de colas compartido			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
suceso ssl			MQIA_SSL_EVENT	MQINQ
Fips ssl			MQIA_SSL_FIPS_REQUIRED	MQINQ
Cuenta restablecimiento clave SSL			MQIA_SSL_RESET_COUNT	MQINQ
suceso de inicio-detención			MQIA_START_STOP_EVENT	MQINQ
Intervalo de estadísticas			MQIA_INTERVALO_ESTADÍSTICAS	MQINQ
Disponibilidad de punto de sincronismo			MQIA_SYNCPOINT	MQINQ
canales tcp			MQIA_TCP_CANALES	MQINQ
Keep alive de TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nombre TCP			MQCA_TCP_NAME	MQINQ
Tipo de pila TCP			MQIA_TCP_STACK_TYPE	MQINQ
Registro de la ruta de rastreo			MQIA_GRABACIÓN_RUTA_RASTREO	MQINQ
Activar intervalo			MQIA_INTERVALO_DESENCADENANTE	MQINQ
ID de usuario	MQCSP	CSPUserIdPtr		MQCONNX
	MQCSP	CSPUserIdDesplazamiento		MQCONNX
	MQCSP	CSPUserIdLongitud		MQCONNX

ImqReferenceReferencia cruzada de cabecera

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 864. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
entorno de destino	MQRMH	DestEnvLongitud, DestEnvDesplazamiento
nombre de destino	MQRMH	DestNameLongitud, DestNameDesplazamiento
ID de instancia	MQRMH	ObjectInstanceId
longitud lógica	MQRMH	DataLogicalLength
desplazamiento lógico	MQRMH	DataLogicalOffset
desplazamiento lógico 2	MQRMH	DataLogicalOffset2
Tipo de referencia	MQRMH	ObjectType
Entorno de origen	MQRMH	SrcEnvLongitud, SrcEnvDesplazamiento
Nombre del origen	MQRMH	SrcNameLongitud, SrcNameDesplazamiento

Referencia cruzada de ImqTrigger

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 865. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
ID de aplicación	MQTM	ApplId
Tipo de aplicación	MQTM	ApplType
Datos de entorno	MQTM	EnvData
nombre de proceso	MQTM	ProcessName
nombre de cola	MQTM	QName
Datos desencadenantes	MQTM	TriggerData
datos de usuario	MQTM	UserData

Referencia cruzada de cabecera ImqWork

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 866. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
token de mensaje	MQWIH	MessageToken
nombre de servicio	MQWIH	ServiceName
paso de servicio	MQWIH	ServiceStep

Clase C++ de registro ImqAuthentication

Esta clase encapsula un registro de información de autenticación (MQAIR) para su uso durante la ejecución del método ImqQueueManager: :connect, para conexiones de cliente TLS personalizadas.

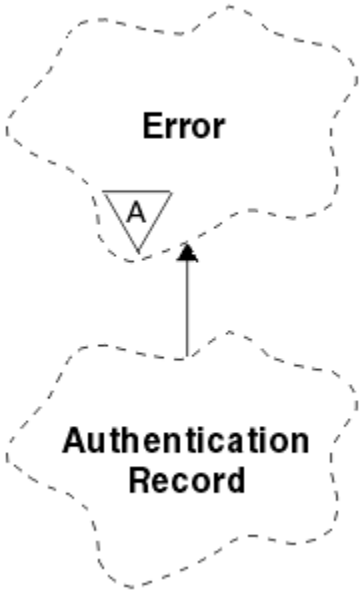


Figura 14. Clase de registro ImqAuthentication

Consulte la descripción del método ImqQueueManager: :connect para obtener más detalles. Esta clase no está disponible en la plataforma z/OS .

- "Atributos de objetos" en la página 1852
- "Constructores" en la página 1853
- "Métodos de objeto (public)" en la página 1853
- "Métodos de objeto (protegidos)" en la página 1853

Atributos de objetos

Nombre de conexión
El nombre de la conexión con el servidor CRL de LDAP. Es la dirección IP o el nombre DNS, seguido opcionalmente por el número de puerto, entre paréntesis.

referencia de conexión
Referencia a un objeto de gestor ImqQueueque proporciona la conexión necesaria a un gestor de colas (local). El valor inicial es cero. No confunda esto con el nombre del gestor de colas que identifica un gestor de colas (posiblemente remoto) para una cola con nombre.

siguiente registro de autenticación
Siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

contraseña
Una contraseña proporcionada para la autenticación de conexión con el servidor CRL de LDAP.

registro de autenticación anterior
Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

tipo
El tipo de información de autenticación contenida en el registro.

nombre de usuario
Un identificador de usuario proporcionado para la autorización al servidor CRL de LDAP.

Constructores

Registro ImqAuthentication();
El constructor predeterminado.

Métodos de objeto (public)

void operator = (const ImqAuthenticationRecord & *aire*);
Copia datos de instancia de *air*, sustituyendo los datos de instancia existentes.

const ImqString & connectionName () Const.
Devuelve el **nombre de conexión**.

void setConnectionName (const ImqString & *nombre*);
Establece el **nombre de conexión**.

void setConnectionName (const char * *nombre* = 0);
Establece el **nombre de conexión**.

ImqQueueManager * connectionReference () Const.
Devuelve la **referencia de conexión**.

void setConnectionReference (ImqQueueManager & *gestor*);
Establece la **referencia de conexión**.

void setConnectionReference (ImqQueueManager * *manager* = 0);
Establece la **referencia de conexión**.

void copyOut (MQAIR * *pAir*);
Copia los datos de instancia en *pAir*, sustituyendo los datos de instancia existentes. Esto puede implicar la asignación de almacenamiento dependiente.

void clear (MQAIR * *pAir*);
Borra la estructura y libera el almacenamiento dependiente al que hace referencia *pAir*.

ImqAuthenticationRegistro * nextAuthenticationRecord () Const.
Devuelve el **siguiente registro de autenticación**.

const ImqString & password () Const.
Devuelve la **contraseña**.

void setPassword (const ImqString & *contraseña*);
Establece la **contraseña**.

void setPassword (const char * *contraseña* = 0);
Establece la **contraseña**.

ImqAuthenticationRegistro * previousAuthenticationRegistro () Const.
Devuelve el **registro de autenticación anterior**.

Tipo MQLONG () Const.
Devuelve el **tipo**.

void setType (const MQLONG *tipo*);
Establece el **tipo**.

const ImqString & userName () Const.
Devuelve el **nombre de usuario**.

void setUserName (const ImqString & *nombre*);
Establece el **nombre de usuario**.

void setUserName (const char * *nombre* = 0);
Establece el **nombre de usuario**.

Métodos de objeto (protegidos)

void setNextAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Establece el **siguiente registro de autenticación**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Establece el **registro de autenticación anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

Clase C++ ImqBinary

Esta clase encapsula una matriz de bytes binarios que se puede utilizar para los valores ImqMessage **señal de contabilidad**, **id de correlación**y **id de mensaje** . Permite una fácil asignación, copia y comparación.

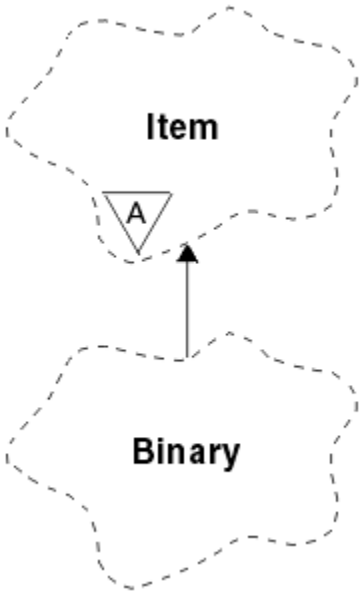


Figura 15. Clase ImqBinary

- “Atributos de objetos” en la página 1854
- “Constructores” en la página 1854
- “Métodos ImqItem sobrecargados” en la página 1855
- “Métodos de objeto (public)” en la página 1855
- “Métodos de objeto (protegidos)” en la página 1855
- “códigos de razón” en la página 1855

Atributos de objetos

datos
Una matriz de bytes de datos binarios. El valor inicial es nulo.

Longitud de datos
Número de bytes. El valor inicial es cero.

puntero de datos
La dirección del primer byte de los **datos**. El valor inicial es cero.

Constructores

ImqBinary( );
El constructor predeterminado.

ImqBinary(const ImqBinary & *binario*);
El constructor de copia.

ImqBinary(const void * *datos*, const size_t *longitud*);
Copia *longitud* bytes de *datos*.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Copia los **datos** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Establece los **datos** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **datos**existentes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqBinary & *binario*);
Copia bytes de *binario*.

ImqBoolean operator == (const ImqBinary & *binario*);
Compara este objeto con *binario*. Devuelve FALSE si no es igual y TRUE de lo contrario. Los objetos son iguales si tienen la misma **longitud de datos** y los bytes coinciden.

ImqBoolean copyOut (void * *buffer*, const size_t *length*, const char *pad* = 0);
Copia hasta *longitud* bytes del **puntero de datos** al *almacenamiento intermedio*. Si la **longitud de datos** es insuficiente, el espacio restante del *almacenamiento intermedio* se rellena con *relleno* bytes. *buffer* puede ser cero si *length* también es cero. *length* no debe ser negativo. Devuelve TRUE si es satisfactorio.

size_t dataLength () const ;
Devuelve la **longitud de datos**.

ImqBoolean setDataLength (const size_t *longitud*);
Establece la **longitud de datos**. Si la **longitud de datos** se cambia como resultado de este método, los datos del objeto no se inicializan. Devuelve TRUE si es satisfactorio.

void * dataPointer () const ;
Devuelve el **puntero de datos**.

ImqBoolean isNull () const ;
Devuelve TRUE si la **longitud de datos** es cero, o si todos los **datos** bytes son cero. De lo contrario, devuelve FALSE.

ImqBoolean set (const void * *buffer*, const size_t *longitud*);
Copia *longitud* bytes del *almacenamiento intermedio*. Devuelve TRUE si es satisfactorio.

Métodos de objeto (protegidos)

void clear ();
Reduce la **longitud de datos** a cero.

códigos de razón

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

Clase C++ ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria.

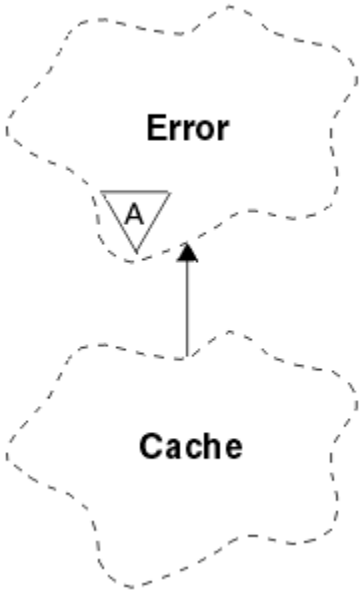


Figura 16. Clase ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria. Puede nombrar un almacenamiento intermedio de memoria de tamaño fijo, o el sistema puede proporcionar una cantidad flexible de memoria automáticamente. Esta clase está relacionada con las llamadas MQI listadas en "Referencia cruzada de ImqCache" en la página 1836.

- "Atributos de objetos" en la página 1856
- "Constructores" en la página 1857
- "Métodos de objeto (public)" en la página 1857
- "códigos de razón" en la página 1858

Atributos de objetos

almacenamiento intermedio automático
Indica si la memoria de almacenamiento intermedio la gestiona automáticamente el sistema (TRUE) o la proporciona el usuario (FALSE). Inicialmente se establece en TRUE.

Este atributo no se ha establecido directamente. Se establece indirectamente utilizando el almacenamiento intermedio **useEmpty** o el método **useFullBuffer** .

Si se proporciona almacenamiento de usuario, este atributo es FALSE, la memoria de almacenamiento intermedio no puede crecer y se pueden producir errores de desbordamiento de almacenamiento intermedio. La dirección y la longitud del almacenamiento intermedio permanecen constantes.

Si no se proporciona almacenamiento de usuario, este atributo es TRUE, y la memoria de almacenamiento intermedio puede crecer de forma incremental para acomodar una cantidad arbitraria de datos de mensaje. Sin embargo, cuando el almacenamiento intermedio crece, la dirección del almacenamiento intermedio puede cambiar, por lo que debe tener cuidado al utilizar el **puntero de almacenamiento intermedio** y el **puntero de datos**.

Longitud de almacenamiento intermedio
Número de bytes de memoria en el almacenamiento intermedio. El valor inicial es cero.

puntero de almacenamiento intermedio
La dirección de la memoria de almacenamiento intermedio. El valor inicial es nulo.

Longitud de datos
El número de bytes que suceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

Desplazamiento de datos
Número de bytes que preceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

puntero de datos
La dirección de la parte del almacenamiento intermedio que se va a escribir o leer en la siguiente. El valor inicial es nulo.

Longitud del mensaje
Número de bytes de datos significativos en el almacenamiento intermedio. El valor inicial es cero.

Constructores

ImqCache( );
El constructor predeterminado.

ImqCache(const ImqCache & *memoria caché*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqCache & *cache*);
Copia hasta **longitud de mensaje** bytes de datos del objeto *cache* en el objeto. Si **almacenamiento intermedio automático** es FALSE, la **longitud de almacenamiento intermedio** ya debe ser suficiente para acomodar los datos copiados.

ImqBoolean automaticBuffer () const ;
Devuelve el valor de **almacenamiento intermedio automático** .

size_t bufferLength () const ;
Devuelve la **longitud de almacenamiento intermedio**.

char * bufferPointer () const ;
Devuelve el **puntero de almacenamiento intermedio**.

void clearMessage ();
Establece la **longitud de mensaje** y el **desplazamiento de datos** en cero.

size_t dataLength () const ;
Devuelve la **longitud de datos**.

size_t dataOffset () const ;
Devuelve el **desplazamiento de datos**.

ImqBoolean setDataOffset (const size_t *desplazamiento*);
Establece el **desplazamiento de datos**. La **longitud de mensaje** se incrementa si es necesario para asegurarse de que no es menor que el **desplazamiento de datos**. Este método devuelve TRUE si es satisfactorio.

char * dataPointer () const ;
Devuelve una copia del **puntero de datos**.

size_t messageLength () const ;
Devuelve la **longitud de mensaje**.

ImqBoolean setMessageLength (const size_t *longitud*);
Establece la **longitud de mensaje**. Aumenta la **longitud del almacenamiento intermedio** si es necesario para asegurarse de que la **longitud del mensaje** no es mayor que la **longitud del almacenamiento intermedio**. Reduce el **desplazamiento de datos** si es necesario para asegurarse de que no es mayor que la **longitud de mensaje**. Devuelve TRUE si es satisfactorio.

ImqBoolean moreBytes (const size_t *bytes-necesario*);
Asegura que *bytes-necesarios* estén disponibles más bytes (para escritura) entre el **puntero de datos** y el final del almacenamiento intermedio. Devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se adquiere más memoria según sea necesario; de lo contrario, la **longitud del almacenamiento intermedio** ya debe ser adecuada.

ImqBoolean read (const size_t *longitud*, char * & *almacenamiento intermedio externo*);
Copia *length* bytes, desde el almacenamiento intermedio que empieza en la posición de **puntero de datos** , en el *external-buffer*. Una vez copiados los datos, el **desplazamiento de datos** se incrementa en *longitud*. Este método devuelve TRUE si es satisfactorio.

ImqBoolean resizeBuffer (const size_t *longitud*);
Varía la **longitud del almacenamiento intermedio**, siempre que el **almacenamiento intermedio automático** sea TRUE. Esto se consigue reasignando la memoria intermedia. Hasta **longitud de mensaje** bytes de datos del almacenamiento intermedio existente se copian en el nuevo. El número máximo copiado es de *longitud* bytes. Se cambia el **puntero de almacenamiento intermedio** . La **longitud de mensaje** y el **desplazamiento de datos** se conservan tan cerca como sea posible dentro de los confines del nuevo almacenamiento intermedio. Devuelve TRUE si es satisfactorio, y FALSE si **almacenamiento intermedio automático** es FALSE.

Nota: Este método puede fallar con MQRC_STORAGE_NOT_AVAILABLE si hay algún problema con los recursos del sistema.

ImqBoolean useEmptyBuffer (const char * *external-buffer*, const size_t *longitud*);
Identifica un almacenamiento intermedio de usuario vacío, estableciendo el **puntero de almacenamiento intermedio** para que apunte a *external-buffer*, la **longitud de almacenamiento intermedio** a *length*y la **longitud de mensaje** a cero. Realiza un **clearMessage**. Si el almacenamiento intermedio está completamente preparado con datos, utilice el método **useFullBuffer** en su lugar. Si el almacenamiento intermedio está parcialmente preparado con datos, utilice el método **setMessageLength** para indicar la cantidad correcta. Este método devuelve TRUE si es satisfactorio.

Este método se puede utilizar para identificar una cantidad fija de memoria, como se ha descrito anteriormente (*almacenamiento intermedio externo* no es nulo y la *longitud* no es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en FALSE, o se puede utilizar para revertir a la memoria flexible gestionada por el sistema (*almacenamiento intermedio externo* es nulo y la *longitud* es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en TRUE.

ImqBoolean useFullBuffer (const char * *externalBuffer*, const size_t *longitud*);
En cuanto a **useEmptyBuffer**, excepto que la **longitud de mensaje** se establece en *length*. Devuelve TRUE si es satisfactorio.

ImqBoolean write (const size_t *longitud*, const char * *búfer externo*);
Copia *longitud* bytes, del *almacenamiento intermedio externo*, en el almacenamiento intermedio empezando en la posición **puntero de datos** . Una vez copiados los datos, el **desplazamiento de datos** se incrementa en *longitud*, y la **longitud de mensaje** se incrementa si es necesario para asegurarse de que no es menor que el nuevo valor de **desplazamiento de datos** . Este método devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se garantiza una cantidad adecuada de memoria; de lo contrario, el **desplazamiento de datos** final no debe superar la **longitud del almacenamiento intermedio**.

códigos de razón

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_TRUNCADO
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_LONGITUD_CERO

Clase C++ ImqChannel

Esta clase encapsula una definición de canal (MQCD) para su uso durante la ejecución del método Manager: :connect, para conexiones de cliente personalizadas.

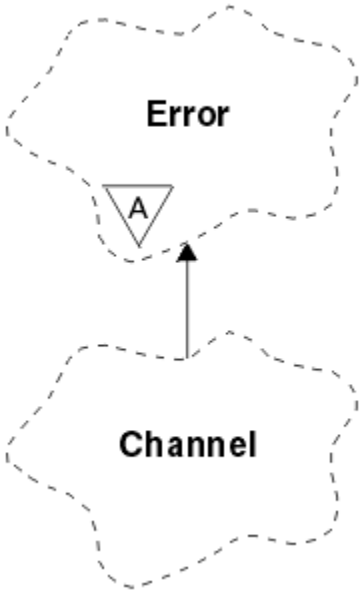


Figura 17. Clase ImqChannel

Consulte la descripción del método Manager: :connect y Programa de ejemplo HELLO WORLD (imqwrld.cpp), para obtener más detalles.

No todos los métodos listados son aplicables a todas las plataformas. Consulte las descripciones de los mandatos DEFINE CHANNEL y ALTER CHANNEL para obtener más información.

La clase ImqChannel no está soportada en z/OS.

- "Atributos de objetos" en la página 1859
- "Constructores" en la página 1860
- "Métodos de objeto (public)" en la página 1861
- "códigos de razón" en la página 1864

Atributos de objetos

latido del corazón por lotes
El número de milisegundos entre comprobaciones de que un canal remoto está activo. El valor inicial es 0.

nombre de canal
El nombre del canal. El valor inicial es nulo.

Nombre de conexión
El nombre de la conexión. Por ejemplo, la dirección IP de un sistema host. El valor inicial es nulo.

Compresión de cabecera
La lista de métodos de compresión de datos de cabecera que el canal admite. Todos los valores iniciales se establecen en MQCOMPRESS_NOT_AVAILABLE.

intervalo de latido cardíaco
Número de segundos entre comprobaciones de que una conexión sigue funcionando. El valor inicial es 300.

Intervalo de mantenimiento activado
Número de segundos pasados a la pila de comunicaciones especificando la temporización de mantener activo para el canal. El valor inicial es MQKAI_AUTO.

dirección local
La dirección de comunicaciones locales para el canal.

longitud máxima del mensaje
Longitud máxima de mensaje soportada por el canal en una sola comunicación. El valor inicial es 4 194 304.

Compresión de mensaje
La lista de métodos de compresión de datos de mensaje que el canal admite. Todos los valores iniciales se establecen en MQCOMPRESS_NOT_AVAILABLE.

nombre modalidad
El nombre de la modalidad. El valor inicial es nulo.

contraseña
Una contraseña proporcionada para la autenticación de conexión. El valor inicial es nulo.

recuento de salidas de recepción
Número de salidas de recepción. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de recepción
Los nombres de las salidas de recepción.

recibir datos de usuario
Datos asociados con salidas de recepción.

Nombre de salida de seguridad
El nombre de una salida de seguridad que se invocará en el lado del servidor de la conexión. El valor inicial es nulo.

datos de usuario de seguridad
Datos que deben pasarse a la salida de seguridad. El valor inicial es nulo.

recuento de salidas de envío
Número de salidas de envío. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de envío
Los nombres de las salidas de envío.

enviar datos de usuario
Datos asociados con salidas de envío.

CipherSpec de SSL
CipherSpec para su uso con TLS.

Tipo de autenticación de cliente SSL
Tipo de autenticación de cliente para utilizar con TLS.

Nombre de igual SSL
Nombre de igual para utilizar con TLS.

nombre del programa de transacciones
El nombre del programa de transacción. El valor inicial es nulo.

tipo de transporte
El tipo de transporte de la conexión. El valor inicial es MQXPT_LU62.

ID de usuario
Identificador de usuario proporcionado para la autorización. El valor inicial es nulo.

Constructores

ImqChannel( ) ;
El constructor predeterminado.

ImqChannel(const ImqChannel & *canal*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqChannel & *canal*);
Copia datos de instancia del *canal*, sustituyendo los datos de instancia existentes.

MQLONG batchHeartBeat () Const.
Devuelve el **latido por lotes**.

ImqBoolean setBatchHeartBeat(const MQLONG *heartbeat* = 0L);
Establece el **latido por lotes**. Este método devuelve TRUE si es satisfactorio.

ImqString channelName() Const.
Devuelve el **nombre de canal**.

ImqBoolean setChannelNombre (const char * *name* = 0);
Establece el **nombre de canal**. Este método devuelve TRUE si es satisfactorio.

ImqString connectionName() Const.
Devuelve el **nombre de conexión**.

ImqBoolean setConnectionNombre (const char * *name* = 0);
Establece el **nombre de conexión**. Este método devuelve TRUE si es satisfactorio.

size_t headerCompressionRecuento () Const.
Devuelve el recuento de técnicas de compresión de datos de cabecera soportadas.

ImqBoolean headerCompression(recuento de const size_t, MQLONG compress []) Const.
Devuelve copias de las técnicas de compresión de datos de cabecera soportadas en **compress**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setHeaderCompression (const size_t count, const MQLONG compress []);

Establece las técnicas de compresión de datos de cabecera soportadas en **compress**.

Establece el recuento de las técnicas de compresión de datos de cabecera soportadas en **recuento**.

Este método devuelve TRUE si es satisfactorio.

Intervalo MQLONG heartBeat() Const.
Devuelve el **intervalo de latido**.

ImqBoolean setHeartBeatInterval(const MQLONG *interval* = 300L);
Establece el **intervalo de latido**. Este método devuelve TRUE si es satisfactorio.

Intervalo MQLONG keepAlive() Const.
Devuelve el **intervalo de mantener activo**.

ImqBoolean setKeepAliveInterval(const MQLONG *interval* = MQKAI_AUTO);
Establece el **intervalo de mantener activo**. Este método devuelve TRUE si es satisfactorio.

ImqString localAddress() const;
Devuelve la **dirección local**.

ImqBoolean setLocalDirección (const char * *address* = 0);
Establece la **dirección local**. Este método devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLongitud () Const.
Devuelve la **longitud máxima de mensaje**.

ImqBoolean setMaximumMessageLength(const MQLONG *length* = 4194304L);
Establece la **longitud máxima de mensaje**. Este método devuelve TRUE si es satisfactorio.

size_t messageCompressionRecuento () Const.
Devuelve el recuento de técnicas de compresión de datos de mensajes soportadas.

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const;
Devuelve copias de las técnicas de compresión de datos de mensaje soportadas en **comprimir**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setMessageCompression (const size_t count, const MQLONG compress []);

Establece las técnicas de compresión de datos de mensaje soportadas para comprimir.

Establece el recuento de las técnicas de compresión de datos de mensajes soportadas.

Este método devuelve TRUE si es satisfactorio.

ImqString modeName() Const.
Devuelve el **nombre de modalidad**.

ImqBoolean setModeNombre (const char * *name* = 0);
Establece el **nombre de modalidad**. Este método devuelve TRUE si es satisfactorio.

ImqString contraseña () Const.
Devuelve la **contraseña**.

ImqBoolean setPassword(const char * *contraseña* = 0);
Establece la **contraseña**. Este método devuelve TRUE si es satisfactorio.

size_t receiveExitRecuento () Const.
Devuelve el **recuento de salidas de recepción**.

ImqString receiveExitNombre ();
Devuelve el primero de los **nombres de salida de recepción**, si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveExitNames (const size_t *recuento*, ImqString * *nombres* []);
Devuelve copias de los **nombres de salida de recepción** en *nombres*. Establece los *nombres* que exceden el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitName(const char * *name* = 0);
Establece los **nombres de salida de recepción** en el único *nombre*. *name* puede estar en blanco o ser nulo. Establece el **recuento de salidas de recepción** en 1 o cero. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *recuento*, const char * *nombres* []);
Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *recuento*, const ImqString * *nombres* []);
Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqString receiveUserData ();
Devuelve el primero de los elementos **recibir datos de usuario** , si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveUserData (const size_t *recuento*, ImqString * *datos* []);
Devuelve copias de los elementos **recibir datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const char * *data* = 0);
Establece los **datos de usuario de recepción** en el elemento único *datos*. Si *data* no es nulo, el **recuento de salidas de recepción** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *recuento*, const char * *datos* []);
Establece los **datos de usuario de recepción** en *datos*. *count* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *recuento*, const ImqString * *datos* []);
Establece los **datos de usuario de recepción** en *datos*. *count* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqString securityExitNombre () Const.
Devuelve el **nombre de salida de seguridad**.

ImqBoolean setSecurityExitName(const char * *name* = 0);
Establece el **nombre de salida de seguridad**. Este método devuelve TRUE si es satisfactorio.

ImqString securityUserData () Const.
Devuelve los **datos de usuario de seguridad**.

ImqBoolean setSecurityUserData(const char * *data* = 0);
Establece los **datos de usuario de seguridad**. Este método devuelve TRUE si es satisfactorio.

size_t sendExitRecuento () Const.
Devuelve el **recuento de salidas de envío**.

ImqString sendExitNombre ();
Devuelve el primero de los **nombres de salida de envío**, si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendExitNames (const size_t *recuento*, ImqString * *nombres* []);
Devuelve copias de los **nombres de salida de envío** en *nombres*. Establece los *nombres* que excedan el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitName(const char * *name* = 0);
Establece los **nombres de salida de envío** en el único *nombre*. *name* puede estar en blanco o ser nulo. Establece el **recuento de salidas de envío** en 1 o cero. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio

ImqBoolean setSendExitNames(const size_t *recuento*, const char * *nombres* []);
Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitNames(const size_t *recuento*, const ImqString * *nombres* []);
Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sendUserDatos ();
Devuelve el primero de los elementos **enviar datos de usuario** , si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendUserData (const size_t *recuento*, ImqString * *datos* []);
Devuelve copias de los elementos **enviar datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const char * *data* = 0);
Establece los **datos de usuario de envío** en los *datos*de un solo elemento. Si *data* no es nulo, el **recuento de salidas de envío** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t *recuento*, const char * *datos* []);
Establece **enviar datos de usuario** en *datos*. *count* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t *recuento*, const ImqString * *datos* []);
Establece **enviar datos de usuario** en *datos*. *count* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sslCipherSpecification () Const.
Devuelve la especificación de cifrado TLS.

ImqBoolean setSslCipherSpecification(const char * *name* = 0);
Establece la especificación de cifrado TLS. Este método devuelve TRUE si es satisfactorio.

Autenticación MQLONG sslClient() Const.
Devuelve el tipo de autenticación de cliente TLS.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);
Establece el tipo de autenticación de cliente TLS. Este método devuelve TRUE si es satisfactorio.

ImqString sslPeerNombre () Const.
Devuelve el nombre de igual TLS.

ImqBoolean setSslPeerName(const char * *name* = 0);
Establece el nombre de igual TLS. Este método devuelve TRUE si es satisfactorio.

ImqString transactionProgramNombre () Const.
Devuelve el **nombre de programa de transacción**.

ImqBoolean setTransactionProgramName(const char * *name* = 0);
Establece el **nombre de programa de transacción**. Este método devuelve TRUE si es satisfactorio.

MQLONG transportType() Const.
Devuelve el **tipo de transporte**.

ImqBoolean setTransportType (const MQLONG *type* = MQXPT_LU62);
Establece el **tipo de transporte**. Este método devuelve TRUE si es satisfactorio.

ImqString userId() Const.
Devuelve el **ID de usuario**.

ImqBoolean setUserId (const char * *id* = 0);
Establece el **ID de usuario**. Este método devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridgeHeader clase C++

Esta clase encapsula características específicas de la estructura de datos MQCIH.

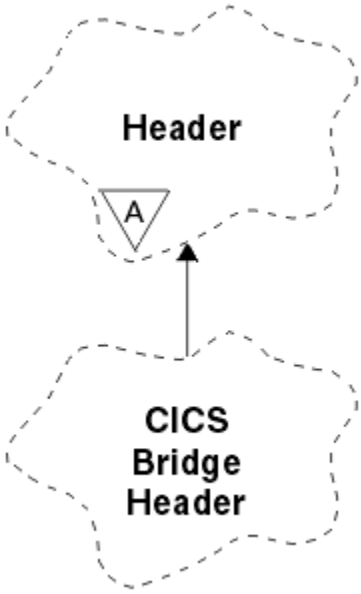


Figura 18. ClaseCICSBridgeHeader de Imq

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes a CICS bridge a través de IBM MQ for z/OS.

- “Atributos de objetos” en la página 1865
- “Constructores” en la página 1867
- “Métodos ImqItem sobrecargados” en la página 1867
- “Métodos de objeto (public)” en la página 1867

- “Datos de objeto (protegidos)” en la página 1870
- “códigos de razón” en la página 1870
- “Códigos de retorno” en la página 1870

Atributos de objetos

Descriptor ADS
Enviar/recibir descriptor ADS. Se establece utilizando MQCADSD_NONE. El valor inicial es MQCADSD_NONE. Son posibles los siguientes valores adicionales:

- MQCADSD_NONE
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

identificador de atención
Tecla AID. El campo debe tener la longitud MQ_ATTENTION_ID_LENGTH.

autenticador
Contraseña o passticket de RACF . El valor inicial contiene espacios en blanco, de longitud MQ_AUTHENTICATOR_LENGTH.

código de terminación anómala de puente
Código de terminación anómala de puente, de longitud MQ_ABEND_CODE_LENGTH. El valor inicial es de cuatro caracteres en blanco. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 867 en la página 1870 para obtener más detalles.

código de cancelación de puente
Código de transacción de terminación anómala de puente. El campo está reservado, debe contener espacios en blanco y tener la longitud MQ_CANCEL_CODE_LENGTH.

código de terminación de puente
Código de terminación, que puede contener el código de terminación IBM MQ o el valor EIBRESP de CICS . El campo tiene el valor inicial de MQCC_OK. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 867 en la página 1870 para obtener más detalles.

desplazamiento de error de puente
Desplazamiento de error de puente. El valor inicial es cero. Este atributo es de sólo lectura.

código de razón de puente
Código de razón. Este campo puede contener la razón IBM MQ o el valor CICS EIBRESP2 . El campo tiene el valor inicial de MQRC_NONE. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 867 en la página 1870 para obtener más detalles.

código de retorno de puente
Código de retorno de CICS bridge. El valor inicial es MQCRC_OK.

tarea conversacional
Si la tarea puede ser conversacional. El valor inicial es MQCCT_NO. Son posibles los siguientes valores adicionales:

- MQCCT_YES
- MQCCT_NO

Posición del cursor
Posición del cursor. El valor inicial es cero.

tiempo de mantenimiento del recurso
Hora de liberación del recurso CICS bridge .

recurso como
Atributo emulado de terminal. El campo debe tener la longitud MQ_FACILITY_LIKE_LENGTH.

señal de recurso
Valor de señal BVT. El campo debe tener la longitud MQ_FACILITY_LENGTH. El valor inicial es MQCFAC_NONE.

función
Función, que puede contener el nombre de llamada IBM MQ o la función EIBFN CICS . El campo tiene el valor inicial de MQCFUNC_NONE, con la longitud MQ_FUNCTION_LENGTH. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 867 en la página 1870 para obtener más detalles.

Los siguientes valores adicionales son posibles cuando la **función** contiene un nombre de llamada IBM MQ :

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

obtener intervalo de espera
Intervalo de espera para una llamada MQGET emitida por la tarea CICS bridge . El valor inicial es MQCGWI_DEFAULT. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST. Son posibles los siguientes valores adicionales:

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

Tipo de enlace
Tipo de enlace. El valor inicial es MQCLT_PROGRAM. Son posibles los siguientes valores adicionales:

- MQCLT_PROGRAM
- TRANSACCIÓN_MQC

identificador de transacción siguiente
ID de la siguiente transacción a adjuntar. El campo debe tener la longitud MQ_TRANSACTION_ID_LENGTH.

longitud de datos de salida
Longitud de datos de COMMAREA. El valor inicial es MQCODL_AS_INPUT.

formato de respuesta
Nombre de formato del mensaje de respuesta. El valor inicial es MQFMT_NONE con la longitud MQ_FORMAT_LENGTH.

código de inicio
Código de inicio de transacción. El campo debe tener la longitud MQ_START_CODE_LENGTH. El valor inicial es MQCSC_NONE. Son posibles los siguientes valores adicionales:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

estado de finalización de tarea
Estado de finalización de tarea. El valor inicial es MQCTES_NOSYNC. Son posibles los siguientes valores adicionales:

- MQCTES_COMMIT
- MQCTES_BACKOUT

- MQCTES_ENDTASK
- MQCTES_NOSYNC

Identificador de transacción
ID de la transacción a adjuntar. El valor inicial debe contener espacios en blanco y debe tener la longitud MQ_TRANSACTION_ID_LENGTH. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

Control UOW
Control de UOW. El valor inicial es MQCUOWC_ONLY. Son posibles los siguientes valores adicionales:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

versión
El número de versión de MQCIH. El valor inicial es MQCIH_VERSION_2. El único otro valor soportado es MQCIH_VERSION_1.

Constructores

ImqCICSBridgeHeader();
El constructor predeterminado.

ImqCICSBridgeHeader(const ImqCICSBridgeHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQCIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato de mensaje en MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQCIH del almacenamiento intermedio de mensajes. Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE. Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqCICSBridgeHeader & *cabecera*);
Copia datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

MQLONG ADSDescriptor () Const.
Devuelve una copia del **descriptor ADS**.

void setADSDescriptor(const MQLONG *descriptor* = MQCADSD_NONE);
Establece el **Descriptor ADS**.

ImqString attentionIdentifier() Const.
Devuelve una copia del **identificador de atención**, rellenado con espacios en blanco finales hasta la longitud MQ_ATTENTION_ID_LENGTH.

void setAttentionIdentifier (const char * *data* = 0);
Establece el **identificador de atención**, rellenado con espacios en blanco finales en la longitud MQ_ATTENTION_ID_LENGTH. Si no se proporcionan *datos* , restablece el **identificador de atención** en el valor inicial.

Autenticador ImqString () Const.
Devuelve una copia del **autenticador**, rellenada con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * *data* = 0);
Establece el **autenticador**, rellenado con espacios en blanco finales en la longitud MQ_AUTHENTICATOR_LENGTH. Si no se proporcionan *datos* , restablece el **autenticador** en el valor inicial.

ImqString bridgeAbendCode () Const.
Devuelve una copia del **código de terminación anómala de puente**, rellenado con espacios en blanco finales hasta la longitud MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelCódigo () Const.
Devuelve una copia del **código de cancelación de puente**, rellenado con espacios en blanco finales hasta la longitud MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char * *data* = 0);
Establece el **código de cancelación de puente**, rellenado con espacios en blanco finales en la longitud MQ_CANCEL_CODE_LENGTH. Si no se proporcionan *datos* , restablece el **código de cancelación de puente** al valor inicial.

MQLONG bridgeCompletionCódigo () Const.
Devuelve una copia del **código de terminación de puente**.

MQLONG bridgeErrorOffset () Const.
Devuelve una copia del **desplazamiento de error de puente**.

MQLONG bridgeReasonCódigo () Const.
Devuelve una copia del **código de razón de puente**.

MQLONG bridgeReturnCódigo () Const.
Devuelve el **código de retorno de puente**.

MQLONG conversationalTask() Const.
Devuelve una copia de la **tarea conversacional**.

void setConversationalTask (const MQLONG *task* = MQCCT_NO);
Establece la **tarea conversacional**.

MQLONG cursorPosition() Const.
Devuelve una copia de la **posición del cursor**.

void setCursorPosition (const MQLONG *position* = 0);
Establece la **posición del cursor**.

MQLONG facilityKeepHora () Const.
Devuelve una copia del **tiempo de mantenimiento del recurso**.

void setFacilityKeepTime(const MQLONG *time* = 0);
Establece el **tiempo de mantenimiento del recurso**.

ImqString facilityLike() Const.
Devuelve una copia del recurso **como**, rellenada con espacios en blanco finales hasta la longitud MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike (const char * *nombre* = 0);
Establece el recurso **como**, rellenado con espacios en blanco finales en la longitud MQ_FACILITY_LIKE_LENGTH. Si no se proporciona ningún *name* , restablece el recurso **como** el valor inicial.

ImqBinary facilityToken() Const.
Devuelve una copia de la **señal de recurso**.

ImqBoolean setFacilityToken (const ImqBinary & *token*);
Establece la **señal de recurso**. La **longitud de datos** de *token* debe ser cero o MQ_FACILITY_LENGTH. Devuelve TRUE si es satisfactorio.

void setFacilityToken (const MQBYTE8 *señal* = 0);
Establece la **señal de recurso**. *token* puede ser cero, que es lo mismo que especificar MQCFAC_NONE. Si *token* es distinto de cero, debe direccionar MQ_FACILITY_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQCFAC_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma. Por ejemplo, (MQBYTE *) MQCFAC_NONE.

Función ImqString () Const.
Devuelve una copia de la **función**, rellenada con espacios en blanco finales hasta la longitud MQ_FUNCTION_LENGTH.

MQLONG getWaitIntervalo () Const.
Devuelve una copia del **intervalo de espera de obtención**.

void setGetWaitInterval(const MQLONG *intervalo* = MQCGWI_DEFA
Establece el **intervalo de espera de obtención**.

MQLONG linkType() Const.
Devuelve una copia del **tipo de enlace**.

void setLinkType (const MQLONG *type* = MQCLT_PROGRAM);
Establece el **tipo de enlace**.

ImqString nextTransactionIdentificador () Const.
Devuelve una copia de los datos del **siguiente identificador de transacción** , rellenados con blancos de cola hasta la longitud MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLongitud () Const.
Devuelve una copia de la **longitud de datos de salida**.

void setOutputDataLength(const MQLONG *length* = MQCODL_AS_INPUT);
Establece la **longitud de datos de salida**.

ImqString replyToFormato () Const.
Devuelve una copia del nombre de **formato de respuesta** , rellenado con blancos de cola hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nombre* = 0);
Establece el **formato de respuesta**, rellenado con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH. Si no se proporciona ningún *nombre* , restablece el **formato de respuesta** al valor inicial.

ImqString startCode() Const.
Devuelve una copia del **código de inicio**, rellenado con espacios en blanco finales hasta la longitud MQ_START_CODE_LENGTH.

void setStartCode (const char * *data* = 0);
Establece los datos del **código de inicio** , rellenados con espacios en blanco finales en la longitud MQ_START_CODE_LENGTH. Si no se proporcionan *datos* , restablece el **código de inicio** en el valor inicial.

MQLONG taskEndEstado () Const.
Devuelve una copia del **estado de finalización de tarea**.

ImqString transactionIdentifier() Const.
Devuelve una copia de los datos del **identificador de transacción** , rellenados con espacios en blanco finales hasta la longitud MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentifier (const char * *data* = 0);
Establece el **identificador de transacción**, rellenado con espacios en blanco finales en la longitud MQ_TRANSACTION_ID_LENGTH. Si no se proporcionan *datos* , restablece el **identificador de transacción** en el valor inicial.

MQLONG UOWControl () Const.
Devuelve una copia del **control de UOW**.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);
Establece el **control de UOW**.

MQLONG versión () Const.
Devuelve el número de **versión** .

ImqBoolean setVersion(const MQLONG *version* = MQCIH_VERSION_2);
Establece el número de **versión** . Devuelve TRUE si es satisfactorio.

Datos de objeto (protegidos)

MQLONG *olVersion*
El número máximo de versión de MQCIH que se puede acomodar en el almacenamiento asignado para *opcih*.

PMQCIH *opcih*
La dirección de una estructura de datos MQCIH. La cantidad de almacenamiento asignado se indica mediante *olVersion*.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- VERSIÓN_ERROR_MQRC

Códigos de retorno

Tabla 867. Códigos de retorno de claseCICSBridgeHeader de Imq

Código de retorno	Función	CompCode	Razón	Código de terminación anómala
MQCRC_OK				
ERROR DE MQCRC_BRIDGE_			MQFB_CICS	
MQCRC_MQ_API_ERROR	Nombre de llamada IBM MQ	IBM MQ CompCode	IBM MQ Razón	
MQCRC_BRIDGE_TIMEOUT	Nombre de llamada IBM MQ	IBM MQ CompCode	IBM MQ Razón	
MQCRC_CICS_ERROR _EXEC_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_SECURITY_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_PROGRAM_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				ABCODE de CICS
MQCRC_APPLICATION_ABEND				ABCODE de CICS

Clase C++ ImqDeadLetterHeader

Esta clase encapsula características de la estructura de datos MQDLH.

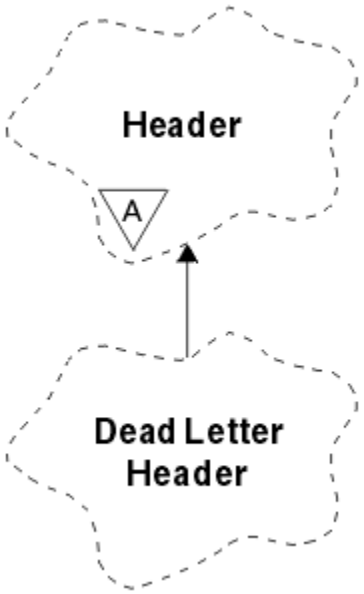


Figura 19. clase ImqDeadLetterHeader

Los objetos de esta clase suelen ser utilizados por una aplicación que encuentra un mensaje que no se puede procesar. Un nuevo mensaje que comprende una cabecera de mensaje no entregado y el contenido del mensaje se coloca en la cola de mensajes no entregados y el mensaje se descarta.

- "Atributos de objetos" en la página 1871
- "Constructores" en la página 1872
- "Métodos ImqItem sobrecargados" en la página 1872
- "Métodos de objeto (public)" en la página 1872
- "Datos de objeto (protegidos)" en la página 1873
- "códigos de razón" en la página 1873

Atributos de objetos

código de razón de mensaje no enviado
La razón por la que el mensaje ha llegado a la cola de mensajes no entregados. El valor inicial es MQRC_NONE.

Nombre del gestor de colas de destino
El nombre del gestor de colas de destino original. El nombre es una serie de longitud MQ_Q_MGR_NAME_LENGTH. Su valor inicial es nulo.

Nombre de la cola de destino
El nombre de la cola de destino original. El nombre es una serie de longitud MQ_Q_NAME_LENGTH. Su valor inicial es nulo.

Nombre de aplicación de transferencia
Nombre de la aplicación que transfiere el mensaje a la cola de mensajes no entregados. El nombre es una serie de longitud MQ_PUT_APPL_NAME_LENGTH. Su valor inicial es nulo.

Tipo de aplicación de transferencia
El tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados. El valor inicial es cero.

Fecha de transferencia
La fecha en la que el mensaje se transfirió a la cola de mensajes no entregados. La fecha es una serie de longitud MQ_PUT_DATE_LENGTH. Su valor inicial es una serie nula.

Hora de transferencia
Hora a la que se transfirió el mensaje a la cola de mensajes no entregados. La hora es una serie de longitud MQ_PUT_TIME_LENGTH. Su valor inicial es una serie nula.

Constructores

ImqDeadLetterHeader( );
El constructor predeterminado.

ImqDeadLetterHeader(const ImqDeadLetterHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Inserta una estructura de datos MQDLH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante. Establece el formato *msg* en MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página "Clase C++ ImqHeader" en la página 1879 para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Lee una estructura de datos MQDLH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato de ImqMessage debe ser MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página "Clase C++ ImqHeader" en la página 1879 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqDeadLetterHeader & *cabecera*);
Copia los datos de instancia desde la *cabecera*, sustituyendo los datos de instancia existentes.

MQLONG deadLetterReasonCode () Const.
Devuelve el código de razón de mensaje no enviado.

void setDeadLetterReasonCode (const MQLONG *razón*);
Establece el código de razón de mensaje no enviado.

ImqString destinationQueueManagerName () Const.
Devuelve el nombre del gestor de colas de destino, despojado de los espacios en blanco finales.

void setDestinationQueueManagerName (const char * *nombre*);
Establece el nombre del gestor de colas de destino. Trunca los datos más largos que MQ_Q_MGR_NAME_LENGTH (48 caracteres).

ImqString destinationQueueNombre () Const.
Devuelve una copia del nombre de cola de destino, despojada de los espacios en blanco finales.

void setDestinationQueueName (const char * *name*);
Establece el nombre de cola de destino. Trunca los datos más largos que MQ_Q_NAME_LENGTH (48 caracteres).

ImqString putApplicationNombre () Const.
Devuelve una copia del nombre de aplicación put, quitada de los espacios en blanco finales.

void setPutApplicationName (const char * *name* = 0);
Establece el nombre de aplicación de colocación. Trunca los datos más largos que MQ_PUT_APPL_NAME_LENGTH (28 caracteres).

MQLONG putApplicationTipo () Const.
Devuelve el tipo de aplicación de colocación.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Establece el tipo de aplicación de colocación.

ImqString putDate () Const.
Devuelve una copia de la fecha de colocación, despojada de los espacios en blanco finales.

void setPutDate (const char * *date* = 0);
Establece la fecha de colocación. Trunca los datos más largos que MQ_PUT_DATE_LENGTH (8 caracteres).

ImqString putTime () Const.
Devuelve una copia del tiempo de colocación, despojada de los espacios en blanco finales.

void setPutTime (const char * *time* = 0);
Establece la hora de colocación. Trunca los datos más largos que MQ_PUT_TIME_LENGTH (8 caracteres).

Datos de objeto (protegidos)

MQDLH *omqdlh*
Estructura de datos MQDLH.

códigos de razón

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_ERROR
- MQRC_ENCODING_ERROR

ImqDistributionListar clase C++

Esta clase encapsula una lista de distribución dinámica que hace referencia a una o más colas con el fin de enviar un mensaje o mensajes a varios destinos.

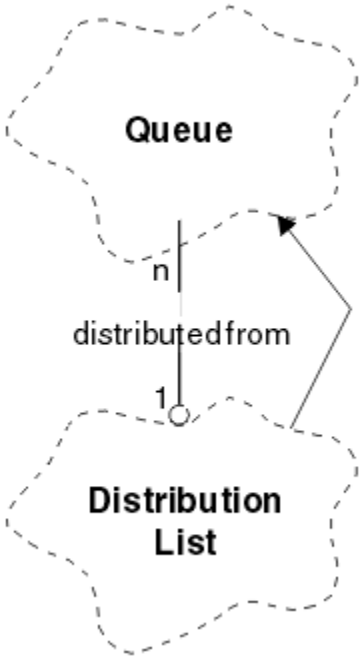


Figura 20. Clase de lista ImqDistribution

- “Atributos de objetos” en la página 1874
- “Constructores” en la página 1874
- “Métodos de objeto (public)” en la página 1874
- “Métodos de objeto (protegidos)” en la página 1874

Atributos de objetos

primera cola distribuida
El primero de uno o más objetos de clase, en ningún orden concreto, en el que la **referencia de lista de distribución** se dirige a este objeto.

Inicialmente no hay tales objetos. Para abrir una lista de ImqDistributioncorrectamente, debe haber al menos un objeto de este tipo.

Nota: Cuando se abre un objeto de lista ImqDistribution, los objetos abiertos que hacen referencia a él se cierran automáticamente.

Constructores

ImqDistributionList ();
El constructor predeterminado.

ImqDistributionList (const ImqDistributionList & *lista*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqDistributionList & *list*);
Se anula la referencia de todos los objetos que hacen referencia a **este** objeto antes de copiarlos. Ningún objeto hará referencia a **este** objeto después de la invocación de este método.

*** firstDistributedQueue () const ;**
Devuelve la **primera cola distribuida**.

Métodos de objeto (protegidos)

void setFirstDistributedQueue (* *queue* = 0);
Establece la **primera cola distribuida**.

Clase C++ ImqError

Esta clase abstracta proporciona información sobre los errores asociados a un objeto.

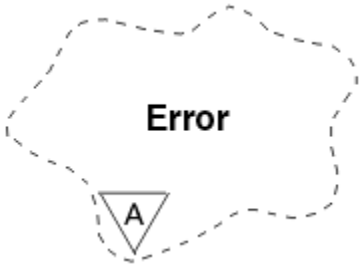


Figura 21. Clase ImqError

- “Atributos de objetos” en la página 1874
- “Constructores” en la página 1875
- “Métodos de objeto (public)” en la página 1875
- “Métodos de objeto (protegidos)” en la página 1875
- “códigos de razón” en la página 1875

Atributos de objetos

código de terminación
El código de terminación más reciente. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

código de razón
El código de razón más reciente. El valor inicial es cero.

Constructores

ImqError( );
El constructor predeterminado.

ImqError(const ImqError & *error*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqError & *error*);
Copia datos de instancia de *error*, sustituyendo los datos de instancia existentes.

void clearErrorCódigos ();
Establece el **código de terminación** y el **código de razón** en cero.

MQLONG completionCode () const ;
Devuelve el **código de terminación**.

MQLONG reasonCode () const ;
Devuelve el **código de razón**.

Métodos de objeto (protegidos)

ImqBoolean checkReadPointer (const void * *pointer*, const size_t *longitud*);
Verifica que la combinación de puntero y longitud es válida para el acceso de sólo lectura y devuelve TRUE si es satisfactoria.

ImqBoolean checkWritePuntero (const void * *pointer*, const size_t *longitud*);
Verifica que la combinación de puntero y longitud es válida para el acceso de lectura-escritura y devuelve TRUE si es satisfactoria.

void setCompletionCode (const MQLONG *code* = 0);
Establece el **código de terminación**.

void setReasonCode (const MQLONG *code* = 0);
Establece el **código de razón**.

códigos de razón

- MQRC_BUFFER_ERROR

Clase C++ ImqGetMessageOptions

Esta clase encapsula la estructura de datos MQGMO

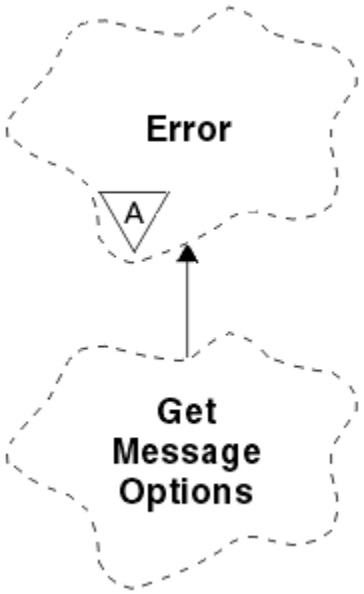


Figura 22. Clase ImqGetMessageOptions

- “Atributos de objetos” en la página 1876
- “Constructores” en la página 1877
- “Métodos de objeto (public)” en la página 1878
- “Métodos de objeto (protegidos)” en la página 1879
- “Datos de objeto (protegidos)” en la página 1879
- “códigos de razón” en la página 1879

Atributos de objetos

Estado de grupo

Estado de un mensaje para un grupo de mensajes. El valor inicial es MQGS_NOT_IN_GROUP. Son posibles los siguientes valores adicionales:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

opciones de coincidencia

Opciones para seleccionar mensajes entrantes. El valor inicial es MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Son posibles los siguientes valores adicionales:

- MQMO_ID_grupo
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

token de mensaje

Señal de mensaje. Un valor binario (MQBYTE16) de longitud MQ_MSG_TOKEN_LENGTH. El valor inicial es MQMTOK_NONE.

opciones

Opciones aplicables a un mensaje. El valor inicial es MQGMO_NO_WAIT. Son posibles los siguientes valores adicionales:

- MQGMO_WAIT
- MQGMO_SYNCPOINT

- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF_QUIESCING
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

Nombre de cola resuelto
Nombre de cola resuelto. Este atributo es de sólo lectura. Los nombres nunca tienen más de 48 caracteres y pueden rellenarse hasta esa longitud con nulos. El valor inicial es una serie nula.

longitud devuelta
Longitud devuelta. El valor inicial es MQRL_UNDEFINED. Este atributo es de sólo lectura.

segmentación
La capacidad de segmentar un mensaje. El valor inicial es MQSEG_INHIBIDO. El valor adicional, MQSEG_ALLOWED, es posible.

estado de segmento
El estado de segmentación de un mensaje. El valor inicial es MQSS_NOT_A_SEGMENT. Son posibles los siguientes valores adicionales:

- SEGMENTO_MQSS
- MQSS_LAST_SEGMENT

participación de punto de sincronismo
TRUE cuando los mensajes se recuperan bajo control de punto de sincronismo.

Intervalo de espera
El tiempo que el método get de clase se detiene mientras espera a que llegue un mensaje adecuado, si todavía no hay uno disponible. El valor inicial es cero, lo que afecta a una espera indefinida. El valor adicional, MQWI_UNLIMITED, es posible. Este atributo se ignora a menos que las opciones incluyan MQGMO_WAIT.

Constructores

ImqGetMessageOptions( );
El constructor predeterminado.

ImqGetMessageOptions(const ImqGetMessageOptions & *gmo*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqGetMessageOptions & *gmo*);
Copia datos de instancia de *gmo*, sustituyendo los datos de instancia existentes.

MQCHAR groupStatus () Const.
Devuelve el estado del grupo.

void setGroupStatus (const MQCHAR *estado*);
Establece el estado del grupo.

MQLONG matchOptions () Const.
Devuelve las opciones de coincidencia.

void setMatchOptions (const MQLONG *opciones*);
Establece las opciones de coincidencia.

ImqBinary messageToken() Const.
Devuelve la señal de mensaje.

ImqBoolean setMessageToken (const ImqBinary & *token*);
Establece la señal de mensaje. La longitud de datos de *token* debe ser cero o MQ_MSG_TOKEN_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setMessageToken (const MQBYTE16 *señal* = 0);
Establece la señal de mensaje. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe dirigirse a MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos, como MQMTOK_NONE, es posible que no sea necesario realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQMTOK_NONE.

Opciones MQLONG () Const.
Devuelve las opciones.

void setOptions (const MQLONG *opciones*);
Establece las opciones, incluido el valor de participación de punto de sincronismo.

ImqString resolvedQueueNombre () Const.
Devuelve una copia del nombre de cola resuelto.

MQLONG returnedLength() Const.
Devuelve la longitud devuelta.

Segmentación MQCHAR () Const.
Devuelve la segmentación.

void setSegmentation (const MQCHAR *valor*);
Establece la segmentación.

MQCHAR segmentStatus () Const.
Devuelve el estado del segmento.

void setSegmentStatus (const MQCHAR *estado*);
Establece el estado del segmento.

ImqBoolean syncPointParticipación () Const.
Devuelve el valor de participación de punto de sincronismo, que es TRUE si las opciones incluyen MQGMO_SYNCPOINT o MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Establece el valor de participación de punto de sincronismo. Si *sync* es TRUE, modifica las opciones para incluir MQGMO_SYNCPOINT y excluir MQGMO_NO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT. Si *sync* es FALSE, modifica las opciones para incluir MQGMO_NO_SYNCPOINT y para excluir MQGMO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval () Const.
Devuelve el intervalo de espera.

void setWaitInterval (const MQLONG *intervalo*);
Establece el intervalo de espera.

Métodos de objeto (protegidos)

static void setVersionSupported (const MQLONG);
Establece la versión MQGMO. El valor predeterminado es MQGMO_VERSION_3.

Datos de objeto (protegidos)

MQGMO *omqgmo*
Una estructura de datos MQGMO Versión 2. Acceda a los campos MQGMO soportados sólo para MQGMO_VERSION_2 .

PMQGMO *opgmo*
La dirección de una estructura de datos MQGMO. El número de versión de esta dirección se indica en *olVersion*. Inspeccione el número de versión antes de acceder a los campos MQGMO, para asegurarse de que están presentes.

MQLONG *olVersion*
El número de versión de la estructura de datos MQGMO direccionado por *opgmo*.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Clase C++ ImqHeader

Esta clase abstracta encapsula características comunes de la estructura de datos MQDLH.

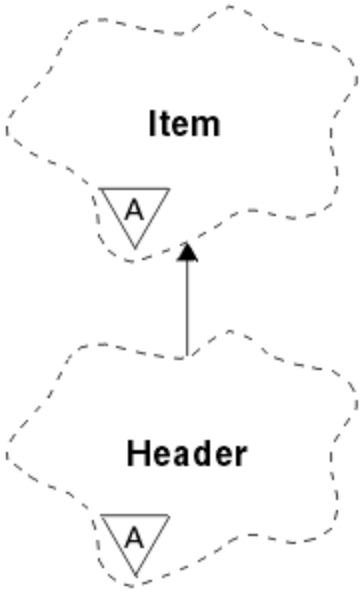


Figura 23. Clase ImqHeader

- “Atributos de objetos” en la página 1879
- “Constructores” en la página 1880
- “Métodos de objeto (public)” en la página 1880

Atributos de objetos

juego de caracteres
Identificador del juego de caracteres codificado original. Inicialmente MQCCSI_Q_MGR.

codificación
La codificación original. Inicialmente MQENC_NATIVE.

formato
El formato original. Inicialmente MQFMT_NONE.

distintivos de cabecera
Los valores iniciales son:

- Cero para objetos de la clase ImqDeadLetterHeader
- MQIIH_NONE para objetos de la claseIMSBridgeHeader de Imq
- MQRMHF_LAST para objetos de la clase de cabecera ImqReference
- MQCIH_NONE para objetos de la claseCICSBridgeHeader de Imq
- MQWIH_NONE para objetos de la clase de cabecera ImqWork

Constructores

ImqHeader( );
El constructor predeterminado.

ImqHeader(const ImqHeader & *cabecera*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqHeader & *cabecera*);
Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

virtual MQLONG characterSet () const ;
Devuelve el **juego de caracteres**.

virtual void setCharacterSet (const MQLONG *ccsid* = MQCCSI_Q_MGR);
Establece el **juego de caracteres**.

Codificación () MQLONG virtual const ;
Devuelve la **codificación**.

virtual void setEncoding (const MQLONG *encoding* = MQENC_NATIVE);
Establece la **codificación**.

virtual ImqString formato () const ;
Devuelve una copia del **formato**, incluidos los espacios en blanco finales.

virtual void setFormat (const char * *name* = 0);
Establece el **formato**, rellenado con 8 caracteres con blancos de cola.

virtual MQLONG headerFlags () const ;
Devuelve los **distintivos de cabecera**.

virtual void setHeaderFlags (const MQLONG *flags* = 0);
Establece los **distintivos de cabecera**.

ImqIMSBridgeHeader clase C++

Esta clase encapsula características de la estructura de datos MQIIH.

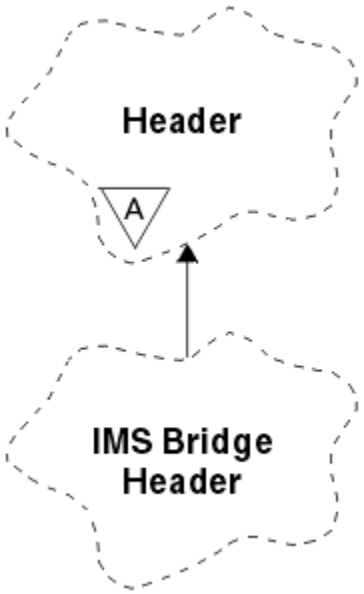


Figura 24. ClaseIMSBridgeHeader de Imq

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes al puente IMS a través de IBM MQ for z/OS.

Nota: El juego de caracteres ImqHeader y la codificación deben tener valores predeterminados y no deben establecerse en ningún otro valor.

- "Atributos de objetos" en la página 1881
- "Constructores" en la página 1882
- "Métodos ImqItem sobrecargados" en la página 1882
- "Métodos de objeto (public)" en la página 1882
- "Datos de objeto (protegidos)" en la página 1883
- "códigos de razón" en la página 1883

Atributos de objetos

autenticador
Contraseña o passticket de RACF , de longitud MQ_AUTHENTICATOR_LENGTH. El valor inicial es MQIAUT_NONE.

Modalidad de confirmación
Modalidad de confirmación. Consulte la publicación *OTMA User's Guide* para obtener más información sobre las modalidades de confirmación de IMS . El valor inicial es MQICM_COMMIT_THEN_SEND. El valor adicional, MQICM_SEND_THEN_COMMIT, es posible.

Alteración temporal del terminal lógico
Alteración temporal de terminal lógico, de longitud MQ_LTERM_OVERRIDE_LENGTH. El valor inicial es una serie nula.

Nombre de la correlación de servicios del formato del mensaje
Nombre de correlación MFS, de longitud MQ_MFS_MAP_NAME_LENGTH. El valor inicial es una serie nula.

formato de respuesta
Formato de cualquier respuesta, de longitud MQ_FORMAT_LENGTH. El valor inicial es MQFMT_NONE.

Ámbito de seguridad
Ámbito del proceso de seguridad de IMS . El valor inicial es MQISS_CHECK. El valor adicional, MQISS_FULL, es posible.

id de instancia de transacción
Identidad de instancia de transacción, un valor binario (MQBYTE16) de longitud MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial es MQITII_NONE.

ESTADO DE TRANSACCIÓN
Estado de la conversación de IMS . El valor inicial es MQITS_NOT_IN_CONVERSATION. El valor adicional, MQITS_IN_CONVERSATION, es posible.

Constructores

ImqIMSBridgeHeader();
El constructor predeterminado.

ImqIMSBridgeHeader(const ImqIMSBridgeHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Inserta una estructura de datos MQIIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensajes existentes más adelante. Establece el formato *msg* en MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Lee una estructura de datos MQIIH del almacenamiento intermedio de mensajes.

Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqIMSBridgeHeader & *cabecera*);
Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

Autenticador ImqString () Const.
Devuelve una copia del autenticador, rellenada con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator (const char * *nombre*);
Establece el autenticador.

MQCHAR commitMode () Const.
Devuelve la modalidad de confirmación.

void setCommitMode (const MQCHAR *mode*);
Establece la modalidad de confirmación.

ImqString logicalTerminalOverride () Const.
Devuelve una copia de la alteración temporal del terminal lógico.

void setLogicalTerminalOverride (const char * *alteración temporal*);
Establece la alteración temporal de terminal lógico.

ImqString messageFormatServicesMapName () Const.
Devuelve una copia del nombre de correlación de servicios de formato de mensaje.

void setMessageFormatServicesMapName (const char * *nombre*);
Establece el nombre de correlación de servicios de formato de mensaje.

ImqString replyToFormato () Const.
Devuelve una copia del formato de respuesta, rellenada con espacios en blanco finales hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat (const char * *format*);
Establece el formato de respuesta, rellenado con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH.

MQCHAR securityScope () Const.
Devuelve el ámbito de seguridad.

void setSecurityScope (const MQCHAR *ámbito*);
Establece el ámbito de seguridad.

ImqBinary transactionInstanceId () Const.
Devuelve una copia del ID de instancia de transacción.

ImqBoolean setTransactionInstanceId (const ImqBinary & *id*);
Establece el ID de instancia de transacción. La longitud de datos de *token* debe ser cero o MQ_TRAN_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setTransactionInstanceId (const MQBYTE16 *id* = 0);
Establece el ID de instancia de transacción. *id* puede ser cero, que es lo mismo que especificar MQITII_NONE. Si *id* es distinto de cero, debe direccionar MQ_TRAN_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQITII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQITII_NONE.

MQCHAR transactionState () Const.
Devuelve el estado de la transacción.

void setTransactionState (const MQCHAR *estado*);
Establece el estado de la transacción.

Datos de objeto (protegidos)

MQIIH *omqiih*
Estructura de datos MQIIH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

Clase C++ ImqItem

Esta clase abstracta representa un elemento, quizás uno de varios, dentro de un mensaje.

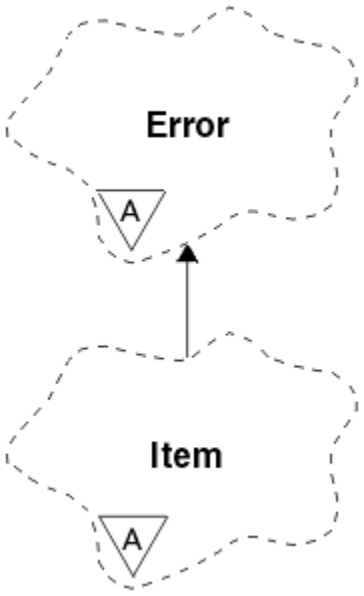


Figura 25. Clase ImqItem

Los elementos se concatenan en un almacenamiento intermedio de mensajes. Cada especialización se asocia con una estructura de datos determinada que empieza con un ID de estructura.

Los métodos polimórficos de esta clase abstracta permiten que los elementos se copien a y desde los mensajes. La clase ImqMessage **readItem** y los métodos **writeItem** proporcionan otro estilo de invocación de estos métodos polimórficos que es más natural para los programas de aplicación.

- "Atributos de objetos" en la página 1884
- "Constructores" en la página 1884
- "Métodos de clase (public)" en la página 1884
- "Métodos de objeto (public)" en la página 1885
- "códigos de razón" en la página 1885

Atributos de objetos

id de estructura
: Una serie de cuatro caracteres al principio de la estructura de datos. Este atributo es de sólo lectura. Considere este atributo para las clases derivadas. No se incluye automáticamente.

Constructores

ImqItem( );
: El constructor predeterminado.

ImqItem(const ImqItem & *elemento*);
: El constructor de copia.

Métodos de clase (public)

static ImqBoolean structureIdIs (const char * *structure-id-to-test*, const ImqMessage & *msg*);
: Devuelve TRUE si el **ID de estructura** del siguiente ImqItem en el *msg* entrante es el mismo que *structure-id-to-test*. El siguiente elemento se identifica como la parte del almacenamiento intermedio de mensajes a la que se dirige actualmente el **puntero de datos** ImqCache . Este método se basa en el **ID de estructura** y, por lo tanto, no se garantiza que funcione para todas las clases derivadas de ImqItem .

Métodos de objeto (public)

void operator = (const ImqItem & *elemento*);
Copia datos de instancia de *elemento*, sustituyendo los datos de instancia existentes.

virtual ImqBoolean copyOut (ImqMessage & *msg*) = 0;
Escribe este objeto como el siguiente elemento en un almacenamiento intermedio de mensajes de salida, añadiéndolo a los elementos existentes. Si la operación de grabación es satisfactoria, aumenta la **longitud de datos** ImqCache . Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

virtual ImqBoolean pasteIn (ImqMessage & *msg*) = 0;
Lee este objeto *de forma destructiva* desde el almacenamiento intermedio de mensajes de entrada. La lectura es destructiva en la que se mueve el **puntero de datos** ImqCache . Sin embargo, el contenido del almacenamiento intermedio sigue siendo el mismo, por lo que los datos se pueden volver a leer restableciendo el **puntero de datos** ImqCache .

La clase (sub) de este objeto debe ser coherente con el **ID de estructura** que se encuentra a continuación en el almacenamiento intermedio de mensajes del objeto *msg* .

La **codificación** del objeto *msg* debe ser MQENC_NATIVE. Se recomienda que los mensajes se recuperen con la **codificación** ImqMessage establecida en MQENC_NATIVE, y con las opciones de ImqGetMessageOptions que incluyen MQGMO_CONVERT.

Si la operación de lectura es satisfactoria, la **longitud de datos** ImqCache se reduce. Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

códigos de razón

- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA

Clase C++ ImqMessage

Esta clase encapsula una estructura de datos MQMD y también maneja la construcción y reconstrucción de datos de mensaje.

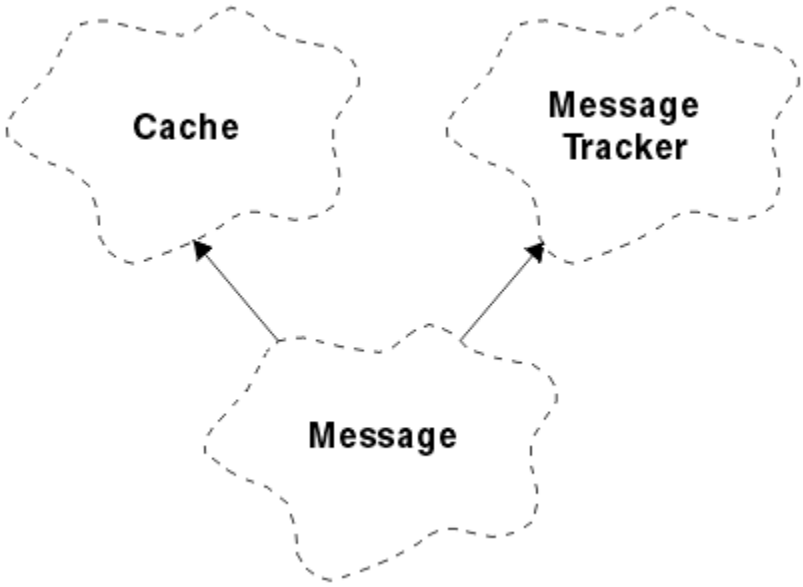


Figura 26. Clase ImqMessage

- “Atributos de objetos” en la página 1886
- “Constructores” en la página 1890
- “Métodos de objeto (public)” en la página 1890
- “Métodos de objeto (protegidos)” en la página 1892
- “Datos de objeto (protegidos)” en la página 1892

Atributos de objetos

Datos de ID de aplicación
Información de identidad asociada a un mensaje. El valor inicial es una serie nula.

Datos de origen de aplicación
Información de origen asociada a un mensaje. El valor inicial es una serie nula.

Recuento de restituciones
El número de veces que un mensaje se ha recuperado provisionalmente y se ha restituido posteriormente. El valor inicial es cero. Este atributo es de sólo lectura.

juego de caracteres
ID de juego de caracteres codificado. El valor inicial es MQCCSI_Q_MGR. Son posibles los siguientes valores adicionales:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

También puede utilizar un ID de juego de caracteres codificado de su elección. Para obtener información sobre esto, consulte “Conversión de páginas de códigos” en la página 956.

codificación
La codificación de máquina de los datos del mensaje. El valor inicial es MQENC_NATIVE.

expiry
Cantidad dependiente del tiempo que controla cuánto tiempo IBM MQ retiene un mensaje no recuperado antes de descartarlo. El valor inicial es MQEI_UNLIMITED.

formato
El nombre del formato (plantilla) que describe el diseño de los datos en el almacenamiento intermedio. Los nombres de más de ocho caracteres se truncan a ocho caracteres. Los nombres siempre se rellenan con espacios en blanco hasta los ocho caracteres. El valor de constante inicial es MQFMT_NONE. Son posibles las siguientes constantes adicionales:

- MQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "Format (MQCHAR8)" en la página 448 del descriptor de mensaje (MQMD).

Distintivos de mensaje
Información de control de segmentación. El valor inicial es MQMF_SEGMENTATION_INHIBIDO. Son posibles los siguientes valores adicionales:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE

tipo de mensaje
La amplia categorización de un mensaje. El valor inicial es MQMT_DATAGRAM. Son posibles los siguientes valores adicionales:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_PRIMERO
- MQMT_APPL_LAST

También puede utilizar un valor específico de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "MsgType (MQLONG)" en la página 438 del descriptor de mensaje (MQMD).

offset
Información de desplazamiento. El valor inicial es cero.

Longitud original
La longitud original de un mensaje segmentado. El valor inicial es MQOL_UNDEFINED.

persistencia
Indica que el mensaje es importante y que se debe realizar una copia de seguridad en todo momento utilizando el almacenamiento persistente. Esta opción implica una penalización del rendimiento. El valor inicial es MQPER_PERSISTENCE_AS_Q_DEF. Son posibles los siguientes valores adicionales:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority
Prioridad relativa para la transmisión y entrega. Los mensajes de la misma prioridad suelen entregarse en la misma secuencia en la que se suministraron (aunque hay varios criterios que deben cumplirse para garantizarlo). El valor inicial es MQPRI_PRIORITY_AS_Q_DEF.

Validación de la propiedad
Especifica si la validación de las propiedades debe tener lugar cuando se establece una propiedad del mensaje. El valor inicial es **MQCMHO_DEFAULT_VALIDATION**. Son posibles los siguientes valores adicionales:

- MQCMHO_VALIDAR
- MQCMHO_NO_VALIDATION

Los métodos siguientes actúan en la **validación de propiedades**:

MQLONG propertyValidation() Const.
Devuelve la opción **property validation** .

void setPropertyValidation (const MQLONG *opción*);
Establece la opción **property validation** .

Nombre de aplicación de transferencia
El nombre de la aplicación que ha colocado un mensaje. El valor inicial es una serie nula.

Tipo de aplicación de transferencia
El tipo de aplicación que ha colocado un mensaje. El valor inicial es MQAT_NO_CONTEXT. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_CICS_PUENTE
- MQAT_DOS
- MQAT_IMS
- MQAT_IMS_BRIDGE
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_QMGR
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT
- MQAT_DESCONOCIDO

- MQAT_USER_FIRST
- MQAT_USER_LAST

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "PutApplType (MQLONG)" en la página 463 del descriptor de mensaje (MQMD).

Fecha de transferencia
La fecha en la que se ha colocado un mensaje. El valor inicial es una serie nula.

Hora de transferencia
Hora a la que se ha colocado un mensaje. El valor inicial es una serie nula.

Nombre de gestor de cola de respuestas
El nombre del gestor de colas al que se debe enviar cualquier respuesta. El valor inicial es una serie nula.

Nombre de cola de respuestas
El nombre de la cola a la que se debe enviar cualquier respuesta. El valor inicial es una serie nula.

informe
Información de comentarios asociada a un mensaje. El valor inicial es MQRO_NONE. Son posibles los siguientes valores adicionales:

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA *
- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA *
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA *
- MQRO_PAN
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NEW_CORREL_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

donde * indica valores que no están soportados en IBM MQ for z/OS.

número de secuencia
Información de secuencia que identifica un mensaje dentro de un grupo. El valor inicial es uno.

longitud total de mensaje
Número de bytes que estaban disponibles durante el intento más reciente de leer un mensaje. Este número será mayor que la **longitud de mensaje** ImqCache si el último mensaje se ha truncado, o si el último mensaje no se ha leído porque se habría producido un truncamiento. Este atributo es de sólo lectura. El valor inicial es cero.

Este atributo puede ser útil en cualquier situación que implique mensajes truncados.

ID de usuario
Identidad de usuario asociada a un mensaje. El valor inicial es una serie nula.

Constructores

ImqMessage( );
El constructor predeterminado.

ImqMessage(const ImqMessage & *msg*);
El constructor de copia. Consulte el método **operator =** para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqMessage & *msg*);
Copia el MQMD y los datos de mensaje de *msg*. Si el usuario ha proporcionado un almacenamiento intermedio para este objeto, la cantidad de datos copiados está restringida al tamaño de almacenamiento intermedio disponible. De lo contrario, el sistema se asegura de que haya disponible un almacenamiento intermedio de tamaño adecuado para los datos copiados.

ImqString applicationIdDatos () const ;
Devuelve una copia de los **datos de ID de aplicación**.

void setApplicationIdData (const char * *data* = 0);
Establece los **datos de ID de aplicación**.

ImqString applicationOriginDatos () const ;
Devuelve una copia de los **datos de origen de aplicación**.

void setApplicationOriginData (const char * *data* = 0);
Establece los **datos de origen de aplicación**.

MQLONG backoutCount () const ;
Devuelve el **recuento de restituciones**.

MQLONG characterSet () const ;
Devuelve el **juego de caracteres**.

void setCharacterSet (const MQLONG *ccsid* = MQCCSI_Q_MGR);
Establece el **juego de caracteres**.

MQLONG codificación () const ;
Devuelve la **codificación**.

void setEncoding (const MQLONG *encoding* = MQENC_NATIVE);
Establece la **codificación**.

MQLONG Caducidad () const ;
Devuelve la **caducidad**.

void setExpiry (const MQLONG *caducidad*);
Establece la **caducidad**.

ImqString formato () const ;
Devuelve una copia del **formato**, incluidos los espacios en blanco finales.

ImqBoolean formatIs (const char * *formato-a-prueba*) const ;
Devuelve TRUE si el formato de es el mismo que el de *format-to-test*.

void setFormat (const char * *name* = 0);
Establece el **formato**, rellenado a ocho caracteres con blancos de cola.

MQLONG messageFlags () const ;
Devuelve los **distintivos de mensaje**.

void setMessageFlags (const MQLONG *distintivos*);
Establece los **distintivos de mensaje**.

MQLONG messageType () const ;
Devuelve el **tipo de mensaje**.

void setMessageType (const MQLONG *tipo*);
Establece el **tipo de mensaje**.

MQLONG Desplazamiento () const ;
Devuelve el **desplazamiento**.

void setOffset (const MQLONG *desplazamiento*);
Establece el **desplazamiento**.

MQLONG originalLength () const ;
Devuelve la **longitud original**.

void setOriginalLength (const MQLONG *longitud*);
Establece la **longitud original**.

MQLONG persistencia () const ;
Devuelve la **persistencia**.

void setPersistence (const MQLONG *persistencia*);
Establece la **persistencia**.

MQLONG prioridad () const ;
Devuelve la **prioridad**.

void setPriority (const MQLONG *prioridad*);
Establece la **prioridad**.

ImqString putApplicationNombre () const ;
Devuelve una copia del **nombre de aplicación de colocación**.

void setPutApplicationName (const char * *name* = 0);
Establece el **nombre de aplicación de colocación**.

MQLONG putApplicationTipo () const ;
Devuelve el **tipo de aplicación put**.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Establece el **tipo de aplicación put**.

ImqString putDate () const ;
Devuelve una copia de la **fecha de colocación**.

void setPutDate (const char * *date* = 0);
Establece la **fecha de colocación**.

ImqString putTime () const ;
Devuelve una copia de la **hora de colocación**.

void setPutTime (const char * *time* = 0);
Establece la **hora de colocación**.

ImqBoolean readItem (ImqItem & *elemento*);
Lee en el objeto *item* del almacenamiento intermedio de mensajes, utilizando el método ImqItem **pasteIn** . Devuelve TRUE si es satisfactorio.

ImqString replyToQueueManagerNombre () const ;
Devuelve una copia del **nombre de gestor de colas de respuesta**.

void setReplyToQueueManagerName (const char * *name* = 0);
Establece el **nombre de gestor de colas de respuesta**.

ImqString replyToQueueName () const ;
Devuelve una copia del **nombre de cola de respuesta**.

void setReplyToQueueName (const char * *name* = 0);
Establece el **nombre de cola de respuesta**.

MQLONG informe () const ;
Devuelve el **informe**.

void setReport (const MQLONG *informe*);
Establece el **informe**.

MQLONG sequenceNumber () const ;
Devuelve el **número de secuencia**.

void setSequenceNumber (const MQLONG *número*);
Establece el **número de secuencia**.

size_t totalMessageLongitud () const ;
Devuelve la **longitud total de mensaje**.

ImqString userId () const ;
Devuelve una copia del **ID de usuario**.

void setUserId (const char * *id* = 0);
Establece el **ID de usuario**.

ImqBoolean writeItem (ImqItem & *elemento*);
Escribe desde el objeto *item* en el almacenamiento intermedio de mensajes, utilizando el método ImqItem **copyOut** . La escritura puede tener la forma de inserción, sustitución o adición: depende de la clase del objeto *item* . Este método devuelve TRUE si es satisfactorio.

Métodos de objeto (protegidos)

static void setVersionSupported (const MQLONG);
Establece la **versión de MQMD**. El valor predeterminado es **MQMD_VERSION_2**.

Datos de objeto (protegidos)

z/OS **MQMD1 *omqmd***
La estructura de datos MQMD en z/OS.

Multi **MQMD2 *omqmd***
La estructura de datos MQMD en Multiplatforms.

Clase C++ de rastreador ImqMessage

Esta clase encapsula los atributos de un objeto ImqMessage o ImqQueue que se pueden asociar con cualquiera de los objetos.

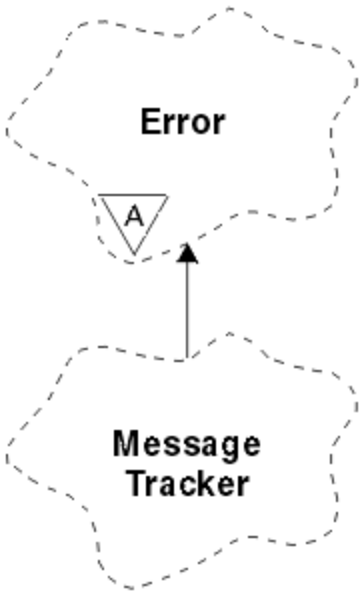


Figura 27. Clase de rastreador ImqMessage

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqMessageTracker” en la página 1841.

- “Atributos de objetos” en la página 1893
- “Constructores” en la página 1894
- “Métodos de objeto (public)” en la página 1894
- “códigos de razón” en la página 1895

Atributos de objetos

Señal de contabilidad
Un valor binario (MQBYTE32) de longitud MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial es MQACT_NONE.

ID de correlación
Un valor binario (MQBYTE24) de longitud MQ_CORREL_ID_LENGTH que se asigna para correlacionar mensajes. El valor inicial es MQCI_NONE. El valor adicional, MQCI_NEW_SESSION, es posible.

información de retorno
Información de retroalimentación que se enviará con un mensaje. El valor inicial es MQFB_NONE. Son posibles los siguientes valores adicionales:

- MQFB_SISTEMA_PRIMERO
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- MQFB_APPL_LAST
- MQFB_COA
- MQFB_COD
- MQFB_EXPIRATION
- MQFB_PAN
- MQFB_NAN
- MQFB_QUIT
- MQFB_DATA_LENGTH_ZERO
- MQFB_DATA_LENGTH_NEGATIVO
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IMS
- MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_BRIDGE_FAILURE
- MQFB_CICS_ERROR _CCSID
- MQFB_CICS_ERROR _CIH
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_ID_CORREL_ERROR
- MQFB_CICS_ERROR _DLQ
- MQFB_CICS_ERROR _ENCODING_
- MQFB_CICS_ERROR DE INTERNAL

- MQFB_CICS_NO_AUTORIZADO
- MQFB_CICS_UOW_BACKED_OUT
- MQFB_CICS_ERROR DE UOW

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "Feedback (MQLONG)" en la página 442 del descriptor de mensaje (MQMD).

ID de grupo
Un valor binario (MQBYTE24) de longitud MQ_GROUP_ID_LENGTH exclusivo dentro de una cola. El valor inicial es MQGI_NONE.

ID de mensaje
Un valor binario (MQBYTE24) de longitud MQ_MSG_ID_LENGTH exclusivo dentro de una cola. El valor inicial es MQMI_NONE.

Constructores

ImqMessageTracker ();
El constructor predeterminado.

ImqMessageTracker (const ImqMessageTracker & *rastreador*);
El constructor de copia. Consulte el método **operator =** para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqMessageTracker & *rastreador*);
Copia datos de instancia del *rastreador*, sustituyendo los datos de instancia existentes.

ImqBinary accountingToken () const ;
Devuelve una copia de la **señal de contabilidad**.

ImqBoolean setAccountingToken (const ImqBinary & *señal*);
Establece la **señal de contabilidad**. La **longitud de datos** de *token* debe ser cero o MQ_ACCOUNTING_TOKEN_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setAccountingToken (const MQBYTE32 *token* = 0);
Establece la **señal de contabilidad**. *token* puede ser cero, que es lo mismo que especificar MQACT_NONE. Si *token* no es cero, debe direccionar MQ_ACCOUNTING_TOKEN_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQACT_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQACT_NONE.

ImqBinary correlationId () const ;
Devuelve una copia del **ID de correlación**.

ImqBoolean setCorrelationId (const ImqBinary & *señal*);
Establece el **ID de correlación**. La **longitud de datos** de *token* debe ser cero o MQ_CORREL_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setCorrelationId (const MQBYTE24 *id* = 0);
Establece el **ID de correlación**. *id* puede ser cero, que es lo mismo que especificar MQCI_NONE. Si *id* es distinto de cero, debe direccionar MQ_CORREL_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQCI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQCI_NONE.

MQLONG comentarios () const ;
Devuelve el **feedback**.

void setFeedback (const MQLONG *feedback*);
Establece los **comentarios**.

ImqBinary groupId () const ;
Devuelve una copia del **id de grupo**.

ImqBoolean setGroupId (const ImqBinary & *señal*);
Establece el **id de grupo**. La **longitud de datos** de *token* debe ser cero o MQ_GROUP_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setGroupId (const MQBYTE24 *id* = 0);
Establece el **id de grupo**. *id* puede ser cero, que es lo mismo que especificar MQGI_NONE. Si *id* es distinto de cero, debe direccionar MQ_GROUP_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQGI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQGI_NONE.

ImqBinary messageId () const ;
Devuelve una copia del **id de mensaje**.

ImqBoolean setMessageId (const ImqBinary & *señal*);
Establece el **id de mensaje**. La **longitud de datos** de *símbolo* debe ser cero o MQ_MSG_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setMessageId (const MQBYTE24 *id* = 0);
Establece el **id de mensaje**. *id* puede ser cero, que es lo mismo que especificar MQMI_NONE. Si *id* es distinto de cero, debe direccionar los bytes MQ_MSG_ID_LENGTH de datos binarios. Cuando se utilizan valores predefinidos como MQMI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQMI_NONE.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Clase C++ ImqNamelist

Esta clase encapsula una lista de nombres.

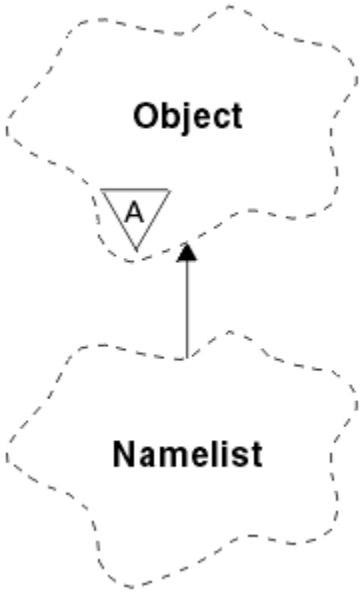


Figura 28. Clase ImqNamelist

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqNamelist” en la página 1841.

- “Atributos de objetos” en la página 1896
- “Constructores” en la página 1896
- “Métodos de objeto (public)” en la página 1896
- “códigos de razón” en la página 1896

Atributos de objetos

Número de nombres
El número de nombres de objeto en **nombres de lista de nombres**. Este atributo es de sólo lectura.

nombres de lista de nombres
Nombres de objeto, cuyo número se indica mediante el **recuento de nombres**. Este atributo es de sólo lectura.

Constructores

ImqNamelist( );
El constructor predeterminado.

ImqNamelist(const ImqNamelist & *lista*);
El constructor de copia. El **estado de apertura** de ImqObject es false.

ImqNamelist(const char * *nombre*);
Establece el nombre de ImqObject en **name**.

Métodos de objeto (public)

void operator = (const ImqNamelist & *lista*);
Copia los datos de instancia de la *lista*, sustituyendo los datos de instancia existentes. El **estado de apertura** de ImqObject es false.

ImqBoolean nameCount(MQLONG & *recuento*);
Proporciona una copia del **recuento de nombres**. Devuelve TRUE si es satisfactorio.

MQLONG nameCount ();
Devuelve el **recuento de nombres** sin ninguna indicación de posibles errores.

ImqBoolean namelistName (const MQLONG *índice*, ImqString & *nombre*);
Proporciona una copia de uno de los **nombres de lista de nombres** por índice basado en cero. Devuelve TRUE si es satisfactorio.

ImqString namelistName (const MQLONG *índice*);
Devuelve uno de los **nombres de lista de nombres** por índice basado en cero sin ninguna indicación de posibles errores.

códigos de razón

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

Clase C++ ImqObject

Esta clase es abstracta. Cuando se destruye un objeto de esta clase, se cierra automáticamente y se interrumpe la conexión del gestor ImqQueue.

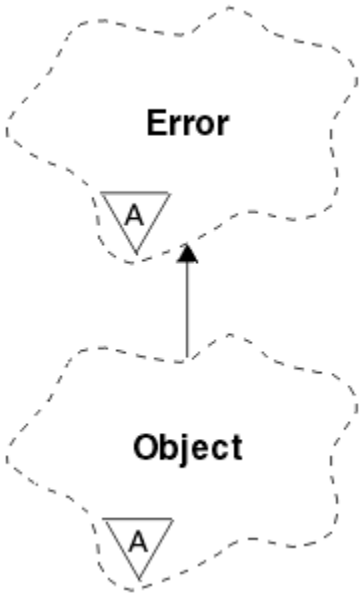


Figura 29. Clase ImqObject

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqObject” en la página 1841.

- “Atributos de clase” en la página 1897
- “Atributos de objetos” en la página 1897
- “Constructores” en la página 1899
- “Métodos de clase (public)” en la página 1899
- “Métodos de objeto (public)” en la página 1899
- “Métodos de objeto (protegidos)” en la página 1901
- “Datos de objeto (protegidos)” en la página 1902
- “códigos de razón” en la página 1902

-

Atributos de clase

comportamiento
Controla el comportamiento de la apertura implícita.

IMQ_IMPL_OPEN (8L)
Se permite la apertura implícita. Éste es el valor predeterminado.

Atributos de objetos

Fecha de modificación
La fecha de modificación. Este atributo es de sólo lectura.

Hora de modificación
El tiempo de alteración. Este atributo es de sólo lectura.

ID de usuario alternativo
El ID de usuario alternativo, con un máximo de MQ_USER_ID_LENGTH caracteres. El valor inicial es una serie nula.

ID de seguridad alternativo
El ID de seguridad alternativo. Un valor binario (MQBYTE40) de longitud MQ_SECURITY_ID_LENGTH. El valor inicial es MQSID_NONE.

opciones de cierre
Opciones que se aplican cuando se cierra un objeto. El valor inicial es MQCO_NONE. Este atributo se ignora durante las operaciones de reapertura implícitas, donde siempre se utiliza un valor de MQCO_NONE.

referencia de conexión
Referencia a un objeto de gestor ImqQueueque proporciona la conexión necesaria a un gestor de colas (local). Para un objeto de gestor ImqQueue, es el propio objeto. El valor inicial es cero.

Nota: No confunda esto con el nombre del gestor de colas que identifica un gestor de colas (posiblemente remoto) para una cola con nombre.

description
Nombre descriptivo (hasta 64 caracteres) del gestor de colas, cola, lista de nombres o proceso. Este atributo es de sólo lectura.

nombre
El nombre (hasta 48 caracteres) del gestor de colas, cola, lista de nombres o proceso. El valor inicial es una serie nula. El nombre de una cola modelo cambia después de un **open** por el nombre de la cola dinámica resultante.

Nota: Un gestor de ImqQueuepuede tener un nombre nulo, que representa el gestor de colas predeterminado. El nombre cambia al gestor de colas real después de una apertura satisfactoria. Una lista de ImqDistributiones dinámica y debe tener un nombre nulo.

siguiente objeto gestionado
Este es el siguiente objeto de esta clase, en ningún orden concreto, que tiene la misma referencia de conexión que este objeto. El valor inicial es cero.

Opciones abiertas
Opciones que se aplican cuando se abre un objeto. El valor inicial es MQOO_INQUIRE. Hay dos formas de establecer los valores adecuados:

1. No establezca las opciones de apertura y no utilice el método de apertura. IBM MQ ajusta automáticamente las opciones de apertura y abre, reabre y cierra automáticamente los objetos según sea necesario. Esto puede dar como resultado operaciones de reapertura innecesarias, porque IBM MQ utiliza el método openFor y esto sólo añade opciones de apertura de forma incremental.
2. Establezca las opciones de apertura antes de utilizar cualquier método que dé como resultado una llamada MQI (consulte “Referencia cruzada de C++ y MQI” en la página 1834). Esto garantiza que no se produzcan operaciones de reapertura innecesarias. Establezca las opciones de apertura de forma explícita si es probable que se produzca alguno de los posibles problemas de reapertura (consulte Reabrir).

   Si utiliza el método abierto, debe asegurarse de que las opciones de apertura son adecuadas en primer lugar. Sin embargo, el uso del método abierto no es obligatorio; IBM MQ sigue mostrando el mismo comportamiento que en el caso 1, pero en esta circunstancia, el comportamiento es eficiente.

Cero no es un valor válido; establezca el valor adecuado antes de intentar abrir el objeto. Esto se puede realizar utilizando **`setOpenOptions`** (*lOpenOptions*) seguido de **open** () o **`openFor`** (*lRequiredOpenOption*).

Nota:

1. MQOO_OUTPUT se sustituye por MQOO_INQUIRE durante el método **open** para una lista de distribución, ya que MQOO_OUTPUT es el único **`open option`** válido en este momento. Sin embargo, se recomienda establecer siempre MQOO_OUTPUT de forma explícita en los programas de aplicación que utilizan el método **open** .
2. Especifique MQOO_RESOLVE_NAMES si desea utilizar los atributos **`resolved queue manager name`** y **`resolved queue name`** de la clase.

estado abierto
Si el objeto está abierto (TRUE) o cerrado (FALSE). El valor inicial es FALSE. Este atributo es de sólo lectura.

objeto gestionado anterior
El objeto anterior de esta clase, en ningún orden concreto, que tiene la misma referencia de conexión que este objeto. El valor inicial es cero.

identificador-gestor-colas
El identificador del gestor de colas. Este atributo es de sólo lectura.

Constructores

ImqObject( );
El constructor predeterminado.

ImqObject(const ImqObject & *objeto*);
El constructor de copia. El estado abierto será FALSE.

Métodos de clase (public)

comportamiento MQLONG estático ();
Devuelve el comportamiento.

void setBehavior(const MQLONG *behavior* = 0);
Establece el comportamiento.

Métodos de objeto (public)

void operator = (const ImqObject & *objeto*);
Realiza un cierre si es necesario y copia los datos de instancia del *objeto*. El estado abierto será FALSE.

ImqBoolean alterationDate(ImqString & *fecha*);
Proporciona una copia de la fecha de modificación. Devuelve TRUE si es satisfactorio.

ImqString alterationDate( );
Devuelve la fecha de modificación sin ninguna indicación de posibles errores.

ImqBoolean alterationTime(ImqString & *hora*);
Proporciona una copia del tiempo de modificación. Devuelve TRUE si es satisfactorio.

ImqString alterationTime( );
Devuelve el tiempo de alteración sin ninguna indicación de posibles errores.

ImqString alternateUserId () Const.
Devuelve una copia del ID de usuario alternativo.

ImqBoolean setAlternateUserId (const char * *id*);
Establece el ID de usuario alternativo. El ID de usuario alternativo sólo se puede establecer mientras el estado abierto es FALSE. Este método devuelve TRUE si es satisfactorio.

ImqBinary alternateSecurityId () Const.
Devuelve una copia del ID de seguridad alternativo.

ImqBoolean setAlternateSecurityId(const ImqBinary & *señal*);
Establece el ID de seguridad alternativo. El ID de seguridad alternativo sólo se puede establecer mientras el estado abierto es FALSE. La longitud de datos de *token* debe ser cero o MQ_SECURITY_ID_LENGTH. Devuelve TRUE si es satisfactorio.

ImqBoolean setAlternateSecurityId(const MQBYTE* *señal* = 0);
Establece el ID de seguridad alternativo. *token* puede ser cero, que es lo mismo que especificar MQSID_NONE. Si *token* es distinto de cero, debe direccionar MQ_SECURITY_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQSID_NONE, es posible que tenga que realizar una conversión para garantizar la coincidencia de firma; por ejemplo, (MQBYTE *) MQSID_NONE.

El ID de seguridad alternativo sólo se puede establecer mientras el estado abierto es TRUE. Devuelve TRUE si es satisfactorio.

ImqBoolean setAlternateSecurityId(const unsigned char * *id* = 0);
Establece el ID de seguridad alternativo.

ImqBoolean close ();
Establece el estado de apertura en FALSE. Devuelve TRUE si es satisfactorio.

MQLONG closeOptions () Const.
Devuelve las opciones de cierre.

void setCloseOptions (const MQLONG *opciones*);
Establece las opciones de cierre.

ImqQueueManager * connectionReference () Const.
Devuelve la referencia de conexión.

void setConnectionReference (ImqQueueManager & *gestor*);
Establece la referencia de conexión.

void setConnectionReference (ImqQueueManager * *manager* = 0);
Establece la referencia de conexión.

descripción ImqBoolean virtual (ImqString & *description*) = 0;
Proporciona una copia de la descripción. Devuelve TRUE si es satisfactorio.

ImqString description ();
Devuelve una copia de la descripción sin ninguna indicación de posibles errores.

nombre ImqBoolean virtual (ImqString & *nombre*);
Proporciona una copia del nombre. Devuelve TRUE si es satisfactorio.

ImqString nombre ();
Devuelve una copia del nombre sin ninguna indicación de posibles errores.

ImqBoolean setName (const char * *name* = 0);
Establece el nombre. El nombre sólo se puede establecer mientras el estado de apertura es FALSE y, para un gestor ImqQueue, mientras el estado de conexión es FALSE. Devuelve TRUE si es satisfactorio.

ImqObject * nextManagedObject () Const.
Devuelve el siguiente objeto gestionado.

ImqBoolean open ();
Cambia el estado de apertura a TRUE abriendo el objeto según sea necesario, utilizando entre otros atributos las opciones de apertura y el nombre. Este método utiliza la información de referencia de conexión y el método de conexión del gestor ImqQueuesi es necesario para asegurarse de que el estado de conexión del gestor ImqQueuees TRUE. Devuelve el estado abierto.

ImqBoolean openFor (const MQLONG *opciones-necesarias* = 0);
Intenta asegurarse de que el objeto está abierto con opciones de apertura, o con opciones de apertura que garantizan el comportamiento implícito en el valor del parámetro *required-options* .

Si *opciones-necesarias* es cero, la entrada es necesaria y cualquier opción de entrada es suficiente. Por lo tanto, si las opciones de apertura ya contienen una de las siguientes:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

las opciones de apertura ya son satisfactorias y no se modifican; si las opciones de apertura todavía no contienen ninguna de estas opciones, MQOO_INPUT_AS_Q_DEF se establece en las opciones de apertura.

Si *required-options* es distinto de cero, las opciones necesarias se añaden a las opciones de apertura; si *required-options* es cualquiera de estas opciones, las demás se restablecerán.

Si se cambia alguna de las opciones de apertura y el objeto ya está abierto, el objeto se cierra temporalmente y se vuelve a abrir para ajustar las opciones de apertura.

Devuelve TRUE si es satisfactorio. El éxito indica que el objeto está abierto con las opciones adecuadas.

MQLONG openOptions () Const.
Devuelve las opciones de apertura.

ImqBoolean setOpenOptions (const MQLONG *opciones*);
Establece las opciones de apertura. Las opciones de apertura sólo se pueden establecer mientras el estado de apertura es FALSE. Devuelve TRUE si es satisfactorio.

ImqBoolean openStatus () Const.
Devuelve el estado abierto.

ImqObject * previousManagedObject () Const.
Devuelve el objeto gestionado anterior.

ImqBoolean queueManagerIdentificador (ImqString & *id*);
Proporciona una copia del identificador del gestor de colas. Devuelve TRUE si es satisfactorio.

ImqString queueManagerIdentificador ();
Devuelve el identificador del gestor de colas sin ninguna indicación de posibles errores.

Métodos de objeto (protegidos)

virtual ImqBoolean closeTemporarily ();
Cierra un objeto de forma segura antes de volver a abrirlo. Devuelve TRUE si es satisfactorio. Este método presupone que el estado abierto es TRUE.

MQHCONN connectionHandle () Const.
Devuelve el MQHCONN asociado a la referencia de conexión. Este valor es cero si no hay ninguna referencia de conexión o si el gestor no está conectado.

ImqBoolean inquire (const MQLONG *int-attr*, MQLONG & *valor*);
Devuelve un valor entero, cuyo índice es un valor MQIA_ *. En caso de error, el valor se establece en MQIAV_UNDEFINED.

ImqBoolean inquire (const MQLONG *atr-car*, char * & *buffer*, const size_t *longitud*);
Devuelve una serie de caracteres, cuyo índice es un valor MQCA_ *.

Nota: Ambos métodos sólo devuelven un único valor de atributo. Si se necesita una *instantánea* de más de un valor, donde los valores son coherentes entre sí durante un instante, IBM MQ C++ no proporciona este recurso y debe utilizar la llamada MQINQ con los parámetros adecuados.

virtual void openInformationDisperse ();
Dispersa información de la sección de variables de la estructura de datos MQOD inmediatamente después de una llamada MQOPEN.

virtual ImqBoolean openInformationPrepare ();
Prepara información para la sección de variables de la estructura de datos MQOD inmediatamente antes de una llamada MQOPEN y devuelve TRUE si es satisfactorio.

ImqBoolean set (const MQLONG *int-attr*, const MQLONG *valor*);
Establece un atributo de entero IBM MQ .

ImqBoolean set (const MQLONG *char-attr*, const char * *buffer*, const size_t *longitud necesaria*);
Establece un atributo de carácter IBM MQ .

void setNextManagedObject (const ImqObject * *objeto* = 0);
Establece el siguiente objeto gestionado.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de objetos gestionados.

void setPreviousManagedObject (const ImqObject * *objeto* = 0);
Establece el objeto gestionado anterior.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de objetos gestionados.

Datos de objeto (protegidos)

MQHOBJ *ohobj*
El descriptor de contexto del objeto IBM MQ (sólo es válido cuando el estado abierto es TRUE).

MQOD *po_omq*
La estructura de datos MQOD incorporada. La cantidad de almacenamiento asignado para esta estructura de datos es la necesaria para una MQOD Versión 2. Inspeccione el número de versión *(omqod.Version)* y acceda a los otros campos como se indica a continuación:

MQOD_VERSION_1
Se puede acceder a todos los demás campos de *omqod* .

MQOD_VERSION_2
Se puede acceder a todos los demás campos de *omqod* .

MQOD_VERSION_3
omqod.pmqod es un puntero a un MQOD más grande, asignado dinámicamente. No se puede acceder a ningún otro campo en *omqod* . Se puede acceder a todos los campos direccionado por *omqod.pmqod* .

Nota: *omqod.pmqod.Version* puede ser menor que *omqod.Version*, lo que indica que IBM MQ MQI client tiene más funcionalidad que el servidor IBM MQ .

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (códigos de razón de MQCLOSE)
- (códigos de razón de MQCONN)
- (códigos de razón de MQINQ)
- (códigos de razón de MQOPEN)
- (códigos de razón de MQSET)

Clase C++ ImqProcess

Esta clase encapsula un proceso de aplicación (un objeto IBM MQ de tipo MQOT_PROCESS) que puede desencadenar un supervisor desencadenante.

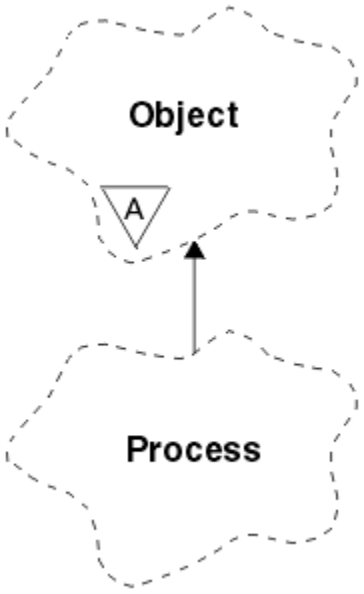


Figura 30. Clase ImqProcess

- “Atributos de objetos” en la página 1903
- “Constructores” en la página 1903
- “Métodos de objeto (public)” en la página 1903

Atributos de objetos

ID de aplicación
La identidad del proceso de aplicación. Este atributo es de sólo lectura.

Tipo de aplicación
El tipo del proceso de aplicación. Este atributo es de sólo lectura.

Datos de entorno
La información de entorno para el proceso. Este atributo es de sólo lectura.

datos de usuario
Datos de usuario para el proceso. Este atributo es de sólo lectura.

Constructores

ImqProcess( );
El constructor predeterminado.

ImqProcess(const ImqProcess & *proceso*);
El constructor de copia. El **estado abierto** de ImqObject es FALSE.

ImqProcess(const char * *nombre*);
Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqProcess & *proceso*);
Realiza un cierre si es necesario y, a continuación, copia los datos de instancia del *proceso*. El **estado abierto** de ImqObject será FALSE.

ImqBoolean applicationId (ImqString & *id*);
Proporciona una copia del **ID de aplicación**. Devuelve TRUE si es satisfactorio.

ImqString applicationId ( );
Devuelve el **ID de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean applicationType (MQLONG & *tipo*);
Proporciona una copia del **tipo de aplicación**. Devuelve TRUE si es satisfactorio.

MQLONG applicationType ();
Devuelve el **tipo de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean environmentData (ImqString & *datos*);
Proporciona una copia de los **datos de entorno**. Devuelve TRUE si es satisfactorio.

ImqString environmentData ( );
Devuelve los **datos de entorno** sin ninguna indicación de posibles errores.

ImqBoolean userData (ImqString & *datos*);
Proporciona una copia de los **datos de usuario**. Devuelve TRUE si es satisfactorio.

ImqString userData ( );
Devuelve los **datos de usuario** sin ninguna indicación de posibles errores.

Clase C++ ImqPutMessageOptions

Esta clase encapsula la estructura de datos MQPMO.

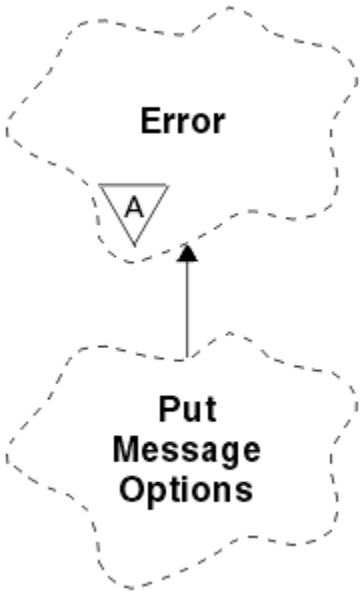


Figura 31. Clase ImqPutMessageOptions

- “Atributos de objetos” en la página 1904
- “Constructores” en la página 1905
- “Métodos de objeto (public)” en la página 1905
- “Datos de objeto (protegidos)” en la página 1906
- “códigos de razón” en la página 1906

Atributos de objetos

referencia de contexto
Una ImqQueue que proporciona un contexto para los mensajes. Inicialmente no hay ninguna referencia.

opciones
Las opciones de colocación de mensaje. El valor inicial es MQPMO_NONE. Son posibles los siguientes valores adicionales:

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT

- MQPMO_NEW_MSG_ID
- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF_QUIESCING

campos de registro
Los distintivos que controlan la inclusión de registros de mensajes de colocación cuando se coloca un mensaje. El valor inicial es MQPMRF_NONE. Son posibles los siguientes valores adicionales:

- MQPMRF_MSG_ID
- ID MQPMRF_CORREL_ID
- MQPMRF_ID_grupo
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN

Los atributos del rastreador ImqMessagese toman del objeto para cualquier campo que se especifique. Los atributos del rastreador ImqMessagese toman del objeto ImqMessage para cualquier campo que no se haya especificado.

Nombre del gestor de colas resuelto
Nombre de un gestor de colas de destino determinado durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

Nombre de cola resuelto
Nombre de una cola de destino determinada durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

participación de punto de sincronismo
TRUE cuando los mensajes se ponen bajo control de punto de sincronismo.

Constructores

ImqPutMessageOptions( );
El constructor predeterminado.

ImqPutMessageOptions(const ImqPutMessageOptions & *pmo*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqPutMessageOptions & *pmo*);
Copia datos de instancia de *pmo*, sustituyendo los datos de instancia existentes.

ImqQueue * contextReference () Const.
Devuelve la referencia de contexto.

void setContextReference (const ImqQueue & *cola*);
Establece la referencia de contexto.

void setContextReference (const ImqQueue * *cola* = 0);
Establece la referencia de contexto.

Opciones MQLONG () Const.
Devuelve las opciones.

void setOptions (const MQLONG *opciones*);
Establece las opciones, incluido el valor de participación de punto de sincronismo.

MQLONG recordFields () Const.
Devuelve los campos de registro.

void setRecordFields (const MQLONG *campos*);
Establece los campos de registro.

ImqString resolvedQueueManagerName () Const.
Devuelve una copia del nombre del gestor de colas resuelto.

ImqString resolvedQueueNombre () Const.
Devuelve una copia del nombre de cola resuelto.

ImqBoolean syncPointParticipación () Const.
Devuelve el valor de participación de punto de sincronismo, que es TRUE si las opciones incluyen MQPMO_SYNCPOINT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Establece el valor de participación de punto de sincronismo. Si *sync* es TRUE, las opciones se modifican para incluir MQPMO_SYNCPOINT y para excluir MQPMO_NO_SYNCPOINT. Si *sync* es FALSE, las opciones se modifican para incluir MQPMO_NO_SYNCPOINT y excluir MQPMO_SYNCPOINT.

Datos de objeto (protegidos)

MQPMO *omqpmo*
Estructura de datos MQPMO.

códigos de razón

- MQRC_STORAGE_NOT_AVAILABLE

Clase C++ ImqQueue

Esta clase encapsula una cola de mensajes (un objeto IBM MQ de tipo MQOT_Q).

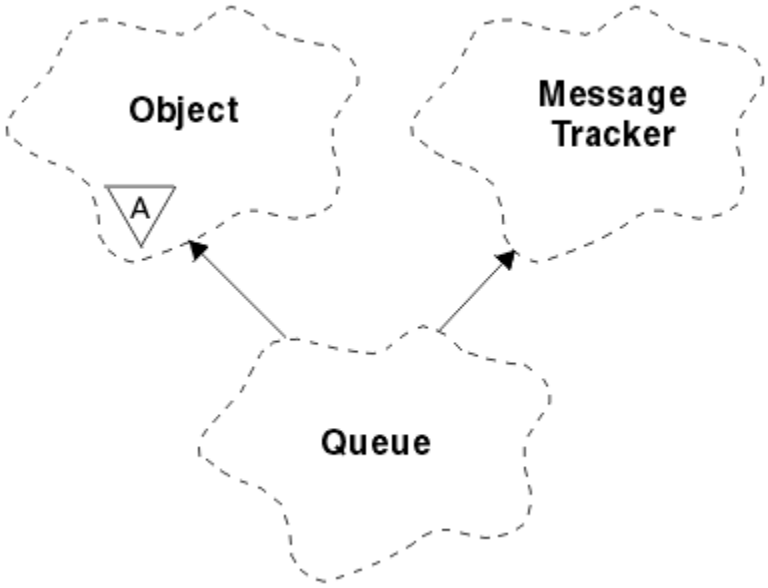


Figura 32. Clase ImqQueue

Esta clase está relacionada con las llamadas MQI listadas en Tabla 862 en la página 1843.

- “Atributos de objetos” en la página 1907
- “Constructores” en la página 1910
- “Métodos de objeto (public)” en la página 1910
- “Métodos de objeto (protegidos)” en la página 1916
- “códigos de razón” en la página 1916

Atributos de objetos

Nombre de reposición en cola para restitución
Nombre de reposición en cola de restitución excesivo. Este atributo es de sólo lectura.

umbral de restituciones
El umbral de restituciones. Este atributo es de sólo lectura.

nombre de cola base
Nombre de la cola en la que se resuelve el alias. Este atributo es de sólo lectura.

Nombre de clúster
Nombre de clúster. Este atributo es de sólo lectura.

Nombre de la lista de nombres del clúster
Nombre de lista de nombres de clúster. Este atributo es de sólo lectura.

Rango de carga trabajo del clúster
El rango de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Prioridad de carga de trabajo del clúster
La prioridad de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster
La carga de trabajo de clúster utiliza el valor de cola. Este atributo es de sólo lectura.

fecha de creación
Datos de creación de cola. Este atributo es de sólo lectura.

Hora de creación
Hora de creación de cola. Este atributo es de sólo lectura.

Profundidad actual
Número de mensajes en la cola. Este atributo es de sólo lectura.

enlace predeterminado
Enlace predeterminado. Este atributo es de sólo lectura.

Opción abierta de entrada predeterminada
Opción de apertura para entrada predeterminada. Este atributo es de sólo lectura.

Persistencia predeterminada
Persistencia de mensajes predeterminada. Este atributo es de sólo lectura.

Prioridad predeterminada
Prioridad de mensaje predeterminada. Este atributo es de sólo lectura.

Tipo de definición
El tipo de definición de la cola. Este atributo es de sólo lectura.

suceso de profundidad alta
Atributo de control para sucesos de profundidad de cola alta. Este atributo es de sólo lectura.

límite alto de profundidad
Límite alto para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad baja
Atributo de control para sucesos de profundidad de cola baja. Este atributo es de sólo lectura.

límite bajo de profundidad
Límite bajo para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad máxima
Atributo de control para sucesos máximos de profundidad de cola. Este atributo es de sólo lectura.

referencia de lista de distribución
Referencia opcional a una lista ImqDistributionque se puede utilizar para distribuir mensajes a más de una cola, incluida ésta. El valor inicial es nulo.

Nota: Cuando se abre un objeto ImqQueue , cualquier objeto de lista ImqDistributionabierto al que haga referencia se cierra automáticamente.

listas de distribución
Capacidad de una cola de transmisión para dar soporte a listas de distribución. Este atributo es de sólo lectura.

Nombre de la cola dinámica
Nombre de cola dinámica. El valor inicial es AMQ.* para todas las plataformas Windows, UNIXy Linux .

Copia en disco de restitución de obtención
Indica si se debe reforzar el recuento de restituciones. Este atributo es de sólo lectura.

Tipo de índice
Tipo de índice. Este atributo es de sólo lectura.

inhibir obtención
Si se permiten operaciones get. El valor inicial depende de la definición de cola. Este atributo sólo es válido para un alias o cola local.

inhibir colocación
Si se permiten las operaciones de colocación. El valor inicial depende de la definición de cola.

Nombre cola iniciación
Nombre de la cola de inicio. Este atributo es de sólo lectura.

Profundidad máxima
Número máximo de mensajes permitidos en la cola. Este atributo es de sólo lectura.

longitud máxima del mensaje
Longitud máxima para cualquier mensaje de esta cola, que puede ser menor que el máximo para cualquier cola gestionada por el gestor de colas asociado. Este atributo es de sólo lectura.

Secuencia de entrega de mensajes
Indica si la prioridad del mensaje es relevante. Este atributo es de sólo lectura.

siguiente cola distribuida
El siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

clase de mensaje no persistente
Nivel de fiabilidad para los mensajes no persistentes colocados en esta cola. Este atributo es de sólo lectura.

Cuenta de entradas abiertas
Número de objetos ImqQueue que están abiertos para entrada. Este atributo es de sólo lectura.

Cuenta de salidas abiertas
Número de objetos ImqQueue que están abiertos para salida. Este atributo es de sólo lectura.

cola distribuida anterior
Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

nombre de proceso
Nombre de la definición de proceso. Este atributo es de sólo lectura.

Contabilidad de la cola
Nivel de información de contabilidad para colas. Este atributo es de sólo lectura.

nombre-gestor-colas
Nombre del gestor de colas (posiblemente remoto) donde reside la cola. No confunda el gestor de colas nombrado aquí con la **referencia de conexión** ImqObject , que hace referencia al gestor de colas (local) que proporciona una conexión. El valor inicial es nulo.

Supervisión de la cola
Nivel de recopilación de datos de supervisión para la cola. Este atributo es de sólo lectura.

estadísticas de cola
Nivel de datos de estadísticas para la cola. Este atributo es de sólo lectura.

Tipo de cola
El tipo de cola. Este atributo es de sólo lectura.

Nombre de gestor de colas remoto
Nombre del gestor de colas remoto. Este atributo es de sólo lectura.

Nombre de cola remota
Nombre de la cola remota tal como se conoce en el gestor de colas remoto. Este atributo es de sólo lectura.

Nombre del gestor de colas resuelto
Nombre de gestor de colas resuelto. Este atributo es de sólo lectura.

Nombre de cola resuelto
Nombre de cola resuelto. Este atributo es de sólo lectura.

Intervalo de retención
Intervalo de retención de cola. Este atributo es de sólo lectura.

ámbito
Ámbito de la definición de cola. Este atributo es de sólo lectura.

intervalo de servicio
Intervalo de servicio. Este atributo es de sólo lectura.

suceso de intervalo de servicio
Atributo de control para sucesos de intervalo de servicio. Este atributo es de sólo lectura.

Posibilidad de compartición
Si la cola puede compartirse. Este atributo es de sólo lectura.

clase de almacenamiento
Clase de almacenamiento. Este atributo es de sólo lectura.

Nombre de cola de transmisión
Nombre de la cola de transmisión. Este atributo es de sólo lectura.

Activar control
Control de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Datos desencadenantes
Datos del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Profundidad de desencadenante
Profundidad del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Prioridad de mensajes desencadenantes
Umbral de prioridad del mensaje para activaciones. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

tipo de activador
Tipo de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

utilización
Uso. Este atributo es de sólo lectura.

Constructores

ImqQueue( );
El constructor predeterminado.

ImqQueue(const ImqQueue & *cola*);
El constructor de copia. El **estado abierto** de ImqObject será FALSE.

ImqQueue(const char * *nombre*);
Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqQueue & *cola*);
Realiza un cierre si es necesario y, a continuación, copia los datos de instancia de la *cola*. El **estado abierto** de ImqObject será FALSE.

ImqBoolean backoutRequeueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de reposición en cola de restitución**. Devuelve TRUE si es satisfactorio.

ImqString backoutRequeueNombre ();
Devuelve el **nombre de reposición en cola de restitución** sin ninguna indicación de posibles errores.

ImqBoolean backoutThreshold (MQLONG & *umbral*);
Proporciona una copia del **umbral de restitución**. Devuelve TRUE si es satisfactorio.

MQLONG backoutThreshold ();
Devuelve el valor de **umbral de restitución** sin ninguna indicación de posibles errores.

ImqBoolean baseQueueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de cola base**. Devuelve TRUE si es satisfactorio.

ImqString baseQueueNombre ();
Devuelve el **nombre de cola base** sin ninguna indicación de posibles errores.

ImqBoolean clusterName(ImqString & *nombre*);
Proporciona una copia del **nombre de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterName( );
Devuelve el **nombre de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterNamelistNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de lista de nombres de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterNamelistNombre ();
Devuelve el **nombre de lista de nombres de clúster** sin ninguna indicación de errores.

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);
Proporciona una copia del valor de prioridad de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadPriority ();
Devuelve el valor de prioridad de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadRank (MQLONG & rango);
Proporciona una copia del valor de rango de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadRank ();
Devuelve el valor de rango de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Proporciona una copia del valor de cola de uso de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadUseQ ();
Devuelve el valor de cola de uso de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean creationDate (ImqString & *fecha*);
Proporciona una copia de la **fecha de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationDate ( );
Devuelve la **fecha de creación** sin ninguna indicación de posibles errores.

ImqBoolean creationTime (ImqString & *hora*);
Proporciona una copia de la **hora de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationTime ( );
Devuelve la **hora de creación** sin ninguna indicación de posibles errores.

ImqBoolean currentDepth (MQLONG & *profundidad*);
Proporciona una copia de la **profundidad actual**. Devuelve TRUE si es satisfactorio.

MQLONG currentDepth ();
Devuelve la **profundidad actual** sin ninguna indicación de posibles errores.

ImqBoolean defaultInputOpenOption (MQLONG & *opción*);
Proporciona una copia de la **opción de apertura de entrada predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultInputOpenOption ();
Devuelve la **opción de apertura de entrada predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPersistence (MQLONG & *persistencia*);
Proporciona una copia de la **persistencia predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPersistence ();
Devuelve la **persistencia predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPriority (MQLONG & *prioridad*);
Proporciona una copia de la **prioridad predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPriority ();
Devuelve la **prioridad predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultBind (MQLONG & *bind*);
Proporciona una copia del **enlace predeterminado**. Devuelve TRUE si es satisfactorio.

MQLONG defaultBind ();
Devuelve el **enlace predeterminado** sin ninguna indicación de posibles errores.

ImqBoolean definitionType (MQLONG & *tipo*);
Proporciona una copia del **tipo de definición**. Devuelve TRUE si es satisfactorio.

MQLONG definitionType ();
Devuelve el **tipo de definición** sin ninguna indicación de posibles errores.

ImqBoolean depthHighEvent (MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad alta**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighSuceso ();
Devuelve el estado de habilitación del **suceso de profundidad alta** sin ninguna indicación de posibles errores.

ImqBoolean depthHighLimit (MQLONG & *límite*);
Proporciona una copia del **límite alto de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighLímite ();
Devuelve el valor de **límite superior de profundidad** sin ninguna indicación de posibles errores.

ImqBoolean depthLowEvent (MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad baja**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowSuceso ();
Devuelve el estado de habilitación del **suceso de profundidad baja** sin ninguna indicación de posibles errores.

ImqBoolean depthLowLimit (MQLONG & *límite*);
Proporciona una copia del **límite bajo de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowLímite ();
Devuelve el valor de **límite inferior de profundidad** sin ninguna indicación de posibles errores.

ImqBoolean depthMaximumEvent (MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG depthMaximumSuceso ();
Devuelve el estado de habilitación del **suceso máximo de profundidad** sin ninguna indicación de posibles errores.

ImqDistributionList * distributionListReference () const ;
Devuelve la **referencia de lista de distribución**.

void setDistributionListReference (ImqDistributionList & *list*);
Establece la **referencia de lista de distribución**.

void setDistributionListReference (ImqDistributionList * *list* = 0);
Establece la **referencia de lista de distribución**.

ImqBoolean distributionLists (Soporte de MQLONG &);
Proporciona una copia del valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

MQLONG distributionLists ();
Devuelve el valor de **listas de distribución** sin ninguna indicación de posibles errores.

ImqBoolean setDistributionLista (const MQLONG *soporte*);
Establece el valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

ImqString dynamicQueueNombre () const ;
Devuelve una copia del **nombre de cola dinámica**.

ImqBoolean setDynamicQueueName (const char * *nombre*);
Establece el **nombre de cola dinámica**. El **nombre de cola dinámica** solo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Devuelve TRUE si es satisfactorio.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *opciones*);
Recupera un mensaje de la cola, utilizando las *opciones*especificadas. Invoca el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** ImqObject incluyen uno de los valores MQOO_INPUT_ * o el valor MQOO_BROWSE, en función de las *opciones*. Si el objeto *msg* tiene un **almacenamiento intermedio automático** ImqCache , el almacenamiento intermedio crece para acomodar cualquier mensaje recuperado. El método **clearMessage** se invoca en el objeto *msg* antes de la recuperación.

Este método devuelve TRUE si es satisfactorio.

Nota: El resultado de la invocación del método es FALSE si el **código de razón** ImqObject es MQRC_TRUNCATED_MSG_FAILED, aunque este **código de razón** se clasifique como aviso. Si se acepta un mensaje truncado, la **longitud de mensaje** ImqCache refleja la longitud truncada. En cualquier caso, la **longitud total de mensaje** ImqMessage indica el número de bytes que estaban disponibles.

ImqBoolean get (ImqMessage & *msg*);
En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *opciones*, const size_t *tamaño-búfer*);
En cuanto a los dos métodos anteriores, excepto que se indica una alteración temporal de *tamaño-búfer* . Si el objeto *msg* utiliza un **almacenamiento intermedio automático** ImqCache , el método **resizeBuffer** se invoca en el objeto *msg* antes de la recuperación de mensajes, y el almacenamiento intermedio no crece más para acomodar ningún mensaje más grande.

ImqBoolean get (ImqMessage & *msg*, const size_t *tamaño-búfer*);
En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBoolean hardenGetBackout (MQLONG & *reforzar*);
Proporciona una copia del valor **harden get backout** . Devuelve TRUE si es satisfactorio.

MQLONG hardenGetBackout ();
Devuelve el valor **harden get backout** sin ninguna indicación de posibles errores.

ImqBoolean indexType(MQLONG & *tipo*);
Proporciona una copia del **tipo de índice**. Devuelve TRUE si es satisfactorio.

MQLONG indexType();
Devuelve el **tipo de índice** sin ninguna indicación de posibles errores.

ImqBoolean inhibitGet (MQLONG & *inhibir*);
Proporciona una copia del valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitGet ();
Devuelve el valor **Impedir obtención** sin ninguna indicación de posibles errores.

ImqBoolean setInhibitGet (const MQLONG *inhibir*);
Establece el valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

ImqBoolean inhibitPut (MQLONG & *inhibir*);
Proporciona una copia del valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitPut ();
Devuelve el valor **Impedir colocación** sin ninguna indicación de posibles errores.

ImqBoolean setInhibitPut (const MQLONG *inhibir*);
Establece el valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

ImqBoolean initiationQueueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de cola de inicio**. Devuelve TRUE si es satisfactorio.

ImqString initiationQueueNombre ();
Devuelve el **nombre de cola de inicio** sin ninguna indicación de posibles errores.

ImqBoolean maximumDepth (MQLONG & *profundidad*);
Proporciona una copia de la **profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG maximumDepth ();
Devuelve la **profundidad máxima** sin ninguna indicación de posibles errores.

ImqBoolean maximumMessageLength (MQLONG & *longitud*);
Proporciona una copia de la **longitud máxima de mensaje**. Devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLongitud ();
Devuelve la **longitud máxima de mensaje** sin ninguna indicación de posibles errores.

ImqBoolean messageDeliverySecuencia (MQLONG & *secuencia*);
Proporciona una copia de la **secuencia de entrega de mensajes**. Devuelve TRUE si es satisfactorio.

MQLONG messageDeliverySecuencia ();
Devuelve el valor de **secuencia de entrega de mensajes** sin ninguna indicación de posibles errores.

ImqQueue * nextDistributedCola () const ;
Devuelve la **siguiente cola distribuida**.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);
Proporciona una copia del valor de clase de mensaje no persistente. Devuelve TRUE si es satisfactorio.

MQLONG nonPersistentMessageClass ();
Devuelve el valor de clase de mensaje no persistente sin ninguna indicación de posibles errores.

ImqBoolean openInputCount (MQLONG & *recuento*);
Proporciona una copia del **recuento de entradas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openInputCount ();
Devuelve el **recuento de entradas abiertas** sin ninguna indicación de posibles errores.

ImqBoolean openOutputCount (MQLONG & *recuento*);
Proporciona una copia del **recuento de salidas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openOutputRecuento ();
Devuelve el **recuento de salidas abiertas** sin ninguna indicación de posibles errores.

ImqQueue * previousDistributedQueue () const ;
Devuelve la **cola distribuida anterior**.

ImqBoolean processName (ImqString & *nombre*);
Proporciona una copia del **nombre de proceso**. Devuelve TRUE si es satisfactorio.

ImqString processName ( );
Devuelve el **nombre de proceso** sin ninguna indicación de posibles errores.

ImqBoolean put (ImqMessage & *msg*);
Coloca un mensaje en la cola, utilizando las opciones de colocación de mensaje predeterminadas. Utiliza el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** de ImqObject incluyen MQOO_OUTPUT.

Este método devuelve TRUE si es satisfactorio.

ImqBoolean put (ImqMessage & *msg*, ImqPutMessageOptions & *pmo*);
Coloca un mensaje en la cola, utilizando el *pmo*especificado. Utiliza el método ImqObject **openFor** según sea necesario para asegurarse de que las **opciones de apertura** de ImqObject incluyen MQOO_OUTPUT y (si las **opciones** de *pmo* incluyen alguno de los valores de MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT correspondientes) MQOO_ * _CONTEXT.

Este método devuelve TRUE si es satisfactorio.

Nota: Si *pmo* incluye una **referencia de contexto**, el objeto referenciado se abre, si es necesario, para proporcionar un contexto.

ImqBoolean queueAccounting (MQLONG & acctq);
Proporciona una copia del valor de contabilidad de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueAccounting ();
Devuelve el valor de contabilidad de cola sin ninguna indicación de posibles errores.

ImqString queueManagerNombre () const ;
Devuelve el **nombre de gestor de colas**.

ImqBoolean setQueueManagerName (const char * *nombre*);
Establece el **nombre de gestor de colas**. El **nombre de gestor de colas** sólo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Este método devuelve TRUE si es satisfactorio.

ImqBoolean queueMonitoring (MQLONG & monq);
Proporciona una copia del valor de supervisión de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueMonitoring ();
Devuelve el valor de supervisión de cola sin ninguna indicación de posibles errores.

ImqBoolean queueStatistics (MQLONG & statq);
Proporciona una copia del valor de estadísticas de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueStatistics ();
Devuelve el valor de estadísticas de cola sin ninguna indicación de posibles errores.

ImqBoolean queueType (MQLONG & *tipo*);
Proporciona una copia del valor de **tipo de cola** . Devuelve TRUE si es satisfactorio.

MQLONG queueType ();
Devuelve el **tipo de cola** sin ninguna indicación de posibles errores.

ImqBoolean remoteQueueManagerName (ImqString & *nombre*);
Proporciona una copia del **nombre de gestor de colas remoto**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueManagerName ( );
Devuelve el **nombre de gestor de colas remoto** sin ninguna indicación de posibles errores.

ImqBoolean remoteQueueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de cola remota**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueNombre ();
Devuelve el **nombre de cola remota** sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueManagerName(ImqString & *nombre*);
Proporciona una copia del **nombre de gestor de colas resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueManagerName( ) ;
Devuelve el **nombre de gestor de colas resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de cola resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueNombre ();
Devuelve el **nombre de cola resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean retentionInterval (MQLONG & *intervalo*);
Proporciona una copia del **intervalo de retención**. Devuelve TRUE si es satisfactorio.

MQLONG retentionInterval ();
Devuelve el **intervalo de retención** sin ninguna indicación de posibles errores.

ImqBoolean scope (MQLONG & *ámbito*);
Proporciona una copia del **ámbito**. Devuelve TRUE si es satisfactorio.

MQLONG Ámbito ();
Devuelve el **ámbito** sin ninguna indicación de posibles errores.

ImqBoolean serviceInterval (MQLONG & *intervalo*);
Proporciona una copia del **intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceInterval ();
Devuelve el **intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBoolean serviceIntervalEvent (MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceIntervalSuceso ();
Devuelve el estado de habilitación del **suceso de intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBoolean compartibilidad (MQLONG & *compartibilidad*);
Proporciona una copia del valor **shareability** . Devuelve TRUE si es satisfactorio.

MQLONG compartibilidad ();
Devuelve el valor de **shareability** sin ninguna indicación de posibles errores.

ImqBoolean storageClass(ImqString & *clase*);
Proporciona una copia de la clase de almacenamiento . Devuelve TRUE si es satisfactorio.

ImqString storageClass( );
Devuelve la **clase de almacenamiento** sin ninguna indicación de posibles errores.

ImqBoolean transmissionQueueNombre (ImqString & *nombre*);
Proporciona una copia del **nombre de cola de transmisión**. Devuelve TRUE si es satisfactorio.

ImqString transmissionQueueNombre ();
Devuelve el **nombre de cola de transmisión** sin ninguna indicación de posibles errores.

ImqBoolean triggerControl (MQLONG & *control*);
Proporciona una copia del valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

MQLONG triggerControl ();
Devuelve el valor de **control de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerControl (const MQLONG *control*);
Establece el valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

ImqBoolean triggerData (ImqString & *datos*);
Proporciona una copia de los **datos desencadenantes**. Devuelve TRUE si es satisfactorio.

ImqString triggerData ( );
Devuelve una copia de los **datos de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerDatos (const char * *datos*);
Establece los **datos de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerDepth (MQLONG & *profundidad*);
Proporciona una copia de la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerDepth ();
Devuelve la **profundidad de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerDepth (const MQLONG *profundidad*);
Establece la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerMessagePriority (MQLONG & *prioridad*);
Proporciona una copia de la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerMessagePriority ();
Devuelve la **prioridad de mensaje desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerMessagePriority (const MQLONG *prioridad*);
Establece la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerType (MQLONG & *tipo*);
Proporciona una copia del **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerType ();
Devuelve el **tipo de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerTipo (const MQLONG *type*);
Establece el **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean uso (MQLONG & *uso*);
Proporciona una copia del valor de **uso** . Devuelve TRUE si es satisfactorio.

MQLONG uso ();
Devuelve el valor de **uso** sin ninguna indicación de posibles errores.

Métodos de objeto (protegidos)

void setNextDistributedQueue (ImqQueue * *queue* = 0);
Establece la **siguiente cola distribuida**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

void setPreviousDistributedQueue (ImqQueue * *queue* = 0);
Establece la **cola distribuida anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- MQRC_CONTEXT_OBJECT_NOT_VALID
- MQRC_CONTEXT_OPEN_ERROR
- MQRC_CURSOR_NOT_VALID
- MQRC_NO_BUFFER

- MQRC_REOPEN_EXCL_INPUT_ERROR
- MQRC_REOPEN_INQUIRE_ERROR
- MQRC_REOPEN_TEMPORARY_Q_ERROR
- (códigos de razón de MQGET)
- (códigos de razón de MQPUT)

Clase C++ del gestor de ImqQueue

Esta clase encapsula un gestor de colas (un objeto IBM MQ de tipo MQOT_Q_MGR).

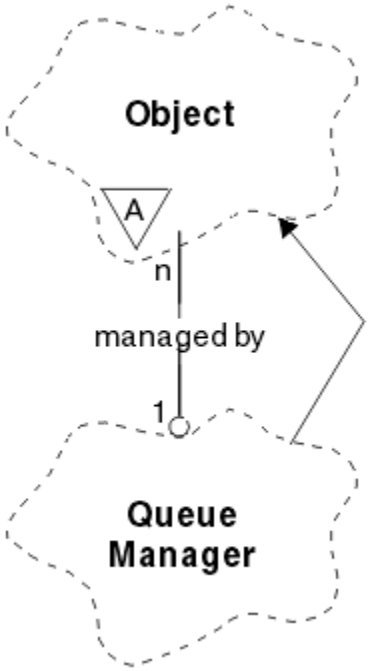


Figura 33. Clase de gestor ImqQueue

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqQueueManager” en la página 1846. No todos los métodos listados son aplicables a todas las plataformas; consulte ALTER QMGR para obtener más detalles.

- “Atributos de clase” en la página 1917
- “Atributos de objetos” en la página 1918
- “Constructores” en la página 1923
- “Destructores” en la página 1923
- “Métodos de clase (public)” en la página 1924
- “Métodos de objeto (public)” en la página 1924
- “Métodos de objeto (protegidos)” en la página 1933
- “Datos de objeto (protegidos)” en la página 1933
- “códigos de razón” en la página 1933

Atributos de clase

comportamiento

Controla el comportamiento de la conexión y desconexión implícitas.

IMQ_EXPL_DISC_BACKOUT (0L)

Una llamada explícita al método de desconexión implica restitución. Este atributo se excluye mutuamente con IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)
Una llamada explícita al método de desconexión implica la confirmación (el valor predeterminado). Este atributo se excluye mutuamente con IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)
Se permite la conexión implícita (el valor predeterminado).

IMQ_IMPL_DISC_BACKOUT (0L)
Una llamada implícita al método de desconexión, que puede producirse durante la destrucción de objetos, implica la restitución. Este atributo se excluye mutuamente con IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)
Una llamada implícita al método de desconexión, que puede producirse durante la destrucción de objetos, implica la confirmación (el valor predeterminado). Este atributo se excluye mutuamente con IMQ_IMPL_DISC_BACKOUT.

En IBM MQ V7.0 y posteriores, las aplicaciones C++ que utilizan una conexión implícita, deben especificar IMQ_IMPL_CONN junto con cualquier otra opción proporcionada en el método `setBehavior()` en un objeto de clase `ImqQueueManager`. Si la aplicación no utiliza el método `setBehavior()` para establecer explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

este cambio no le afecta puesto que MQ_IMPL_CONN está habilitado de forma predeterminada.

Si la aplicación establece explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

debe incluir IMQ_IMPL_CONN en el método `setBehavior()` de la forma siguiente, para permitir que la aplicación complete una conexión implícita:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Atributos de objetos

alteración temporal de conexiones de contabilidad
Permite a las aplicaciones alterar temporalmente el valor de los valores values.This es de sólo lectura.

Intervalo contable
Tiempo antes de que se escriban los registros de contabilidad intermedios (en segundos). Este atributo es de sólo lectura.

registro de actividad
Controla la generación de los informes de actividades. Este atributo es de sólo lectura.

Adoptar nueva comprobación MCA
Los elementos comprobados para determinar si un MCA debe adoptarse cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo. Este atributo es de sólo lectura.

Adoptar nuevo tipo MCA
Si una instancia huérfana de un MCA de un tipo de canal determinado debe reiniciarse automáticamente cuando se detecte una nueva solicitud de canal de entrada que coincida con los nuevos parámetros de comprobación de adopción de mca. Este atributo es de sólo lectura.

Tipo de autenticación
Indica el tipo de autenticación que se está realizando.

suceso de autorización
Controla los sucesos de autorización. Este atributo es de sólo lectura.

Opciones de inicio
Opciones que se aplican al método de inicio. El valor inicial es MQBO_NONE.

suceso de puente
Si se generan sucesos de puente IMS . Este atributo es de sólo lectura.

Definición automática de canal
Valor de definición automática de canal. Este atributo es de sólo lectura.

suceso de definición automática de canal
Valor de suceso de definición automática de canal. Este atributo es de sólo lectura.

Salida de definición automática de canal
Nombre de salida de definición automática de canal. Este atributo es de sólo lectura.

suceso de canal
Indica si se generan sucesos de canal. Este atributo es de sólo lectura.

Adaptadores del iniciador de canal
Número de subtareas de adaptador que se deben utilizar para procesar llamadas de IBM MQ . Este atributo es de sólo lectura.

Control del iniciador de canal
Indica si el iniciador de canal debe iniciarse automáticamente cuando se inicia el gestor de colas. Este atributo es de sólo lectura.

Asignadores de tareas del iniciador de canal
Número de asignadores a utilizar para el iniciador de canal. Este atributo es de sólo lectura.

inicio automático de rastreo de iniciador de canal
Indica si el rastreo de iniciador de canal debe iniciarse automáticamente o no. Este atributo es de sólo lectura.

Tamaño de tabla de rastreo del iniciador de canal
El tamaño del espacio de datos de rastreo del iniciador de canal (en MB). Este atributo es de sólo lectura.

Supervisión de canal
Controla la recopilación de los datos de supervisión para los canales. Este atributo es de sólo lectura.

referencia de canal
Una referencia a una definición de canal para utilizarla durante la conexión de cliente. Mientras está conectado, este atributo se puede establecer en nulo, pero no se puede cambiar a ningún otro valor. El valor inicial es nulo.

Estadísticas del canal
Controla la recopilación de los datos estadísticos para los canales. Este atributo es de sólo lectura.

juego de caracteres
Identificador de juego de caracteres codificado (CCSID). Este atributo es de sólo lectura.

Supervisión de clúster emisor
Controla la recopilación de datos de supervisión en línea para canales emisores de clúster definidos automáticamente. Este atributo es de sólo lectura.

Estadística de clúster emisor
Controla la recopilación de datos estadísticos para los canales emisores de clúster definidos automáticamente. Este atributo es de sólo lectura.

Datos de carga de trabajo de clúster
Datos de salida de carga de trabajo de clúster. Este atributo es de sólo lectura.

salida de carga de trabajo de clúster
Nombre de salida de carga de trabajo de clúster. Este atributo es de sólo lectura.

Longitud de la carga de trabajo de clúster
Longitud de carga de trabajo de clúster. Este atributo es de sólo lectura.

mru de carga de trabajo de clúster
Valor de canales utilizados más recientemente de carga de trabajo de clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster
La carga de trabajo de clúster utiliza el valor de cola. Este atributo es de sólo lectura.

suceso de mandato
Indica si se generan sucesos de mandato. Este atributo es de sólo lectura.

Nombre de cola de entrada de mandatos
Nombre de cola de entrada de mandatos del sistema. Este atributo es de sólo lectura.

Nivel de mandato
Nivel de mandatos soportado por el gestor de colas. Este atributo es de sólo lectura.

Control del servidor de mandatos
Indica si el servidor de mandatos se debe iniciar automáticamente cuando se inicia el gestor de colas. Este atributo es de sólo lectura.

Opciones de conexión
Opciones que se aplican al método de conexión. El valor inicial es MQCNO_NONE. Los siguientes valores adicionales pueden ser posibles, dependiendo de la plataforma:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_HANDLE_SHARE_NONE
- MQCNO_HANDLE_SHARE_BLOCK
- MQCNO_HANDLE_SHARE_NO_BLOCK
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_QSG

ID de conexión
Identificador exclusivo que permite a MQ identificar de forma fiable una aplicación.

estado de conexión
TRUE cuando se conecta al gestor de colas. Este atributo es de sólo lectura.

etiqueta de conexión
Una etiqueta que se va a asociar con una conexión. Este atributo sólo se puede establecer cuando no está conectado. El valor inicial es nulo.

hardware de cifrado
Detalles de configuración para hardware criptográfico. Para conexiones de cliente MQI de MQ .

nombre de cola de mensajes no entregados
Nombre de la cola de mensajes no entregados. Este atributo es de sólo lectura.

nombre de cola de transmisión predeterminada
Nombre de cola de transmisión predeterminado. Este atributo es de sólo lectura.

listas de distribución
Capacidad del gestor de colas para dar soporte a listas de distribución.

grupo dns
El nombre del grupo al que debe unirse el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas cuando se utiliza el soporte de Workload Manager Dynamic Domain Name Services. Este atributo es de sólo lectura.

dns wlm
Indica si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas debe registrarse con el Gestor de carga de trabajo para Dynamic Domain Name Services. Este atributo es de sólo lectura.

primer registro de autenticación
El primero de uno o más objetos de la clase ImqAuthenticationRecord, en ningún orden concreto, en el que la conexión de registro ImqAuthenticationhace referencia a este objeto. Para conexiones de cliente MQI de MQ .

primer objeto gestionado
El primero de uno o más objetos de la clase ImqObject, en ningún orden concreto, en el que la conexión ImqObject hace referencia a este objeto. El valor inicial es cero.

suceso de inhibición
Controla los sucesos de inhibición. Este atributo es de sólo lectura.

Versión de dirección IP
Qué protocolo IP (IPv4 o IPv6) utilizar para una conexión de canal. Este atributo es de sólo lectura.

repositorio de claves
Ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. Para conexiones IBM MQ MQI client .

recuento de restablecimiento de clave
Número de bytes no cifrados enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta. Este atributo sólo se aplica a las conexiones de cliente que utilizan MQCONNX. Véase también recuento de restablecimiento de clave ssl.

Temporizador de escucha
El intervalo de tiempo (en segundos) entre los intentos de IBM MQ de reiniciar el escucha si se ha producido una anomalía de APPC o TCP/IP. Este atributo es de sólo lectura.

suceso local
Controla los sucesos locales. Este atributo es de sólo lectura.

LoggerEvent
Controla si se generan sucesos de registro de recuperación. Este atributo es de sólo lectura.

Nombre de grupo LU
El nombre de LU genérico que debe utilizar el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas. Este atributo es de sólo lectura.

Nombre de LU
El nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 . Este atributo es de sólo lectura.

Sufijo de brazo lu62
El sufijo de SYS1.PARMLIB APPCPMxx, que nomina el LUADD para este iniciador de canal. Este atributo es de sólo lectura.

Canales lu62
El número máximo de canales que pueden ser actuales o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 . Este atributo es de sólo lectura.

máximo de canales activos
Número máximo de canales que pueden estar activos en cualquier momento. Este atributo es de sólo lectura.

Canales máximos
Número máximo de canales que pueden ser actuales (incluidos los canales de conexión de servidor con clientes conectados). Este atributo es de sólo lectura.

máximo de manejadores
Número máximo de descriptores de contexto. Este atributo es de sólo lectura.

longitud máxima del mensaje
Longitud máxima posible para cualquier mensaje en cualquier cola gestionada por este gestor de colas. Este atributo es de sólo lectura.

Prioridad máxima
Prioridad máxima de mensaje. Este atributo es de sólo lectura.

Máx. mensajes no confirmados
Número máximo de mensajes no confirmados dentro de una unidad o trabajo. Este atributo es de sólo lectura.

Contabilidad de MQI
Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI). Este atributo es de sólo lectura.

Estadísticas MQI
Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas. Este atributo es de sólo lectura.

máximo de puerto de salida
El extremo superior del rango de números de puerto que se va a utilizar al enlazar canales de salida. Este atributo es de sólo lectura.

mínimo de puerto de salida
El extremo inferior del rango de números de puerto que se va a utilizar al enlazar canales de salida. Este atributo es de sólo lectura.

contraseña
contraseña asociada con el ID de usuario

suceso de rendimiento
Controla los sucesos de rendimiento. Este atributo es de sólo lectura.

plataforma
Plataforma en la que reside el gestor de colas. Este atributo es de sólo lectura.

Contabilidad de la cola
Controla la recopilación de la información contable para las colas. Este atributo es de sólo lectura.

Supervisión de la cola
Controla la recopilación de los datos de supervisión para las colas. Este atributo es de sólo lectura.

estadísticas de cola
Controla la recopilación de los datos estadísticos para las colas. Este atributo es de sólo lectura.

Tiempo de espera de recepción
Aproximadamente el tiempo que un canal de mensajes TCP/IP esperará a recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo. Este atributo es de sólo lectura.

tiempo de espera de recepción mínimo
El tiempo mínimo que un canal TCP/IP esperará para recibir datos, incluidos los latidos, de su asociado, antes de volver al estado inactivo. Este atributo es de sólo lectura.

Tipo de tiempo de espera de recepción
Se ha aplicado un calificador para recibir el tiempo de espera. Este atributo es de sólo lectura.

suceso remoto
Controla los sucesos remotos. Este atributo es de sólo lectura.

Nombre de depósito
Nombre de repositorio. Este atributo es de sólo lectura.

Lista nombres repositorio
Nombre de lista de nombres de repositorio. Este atributo es de sólo lectura.

nombre de gestor de colas compartido
Si los MQOPENs de una cola compartida donde el nombre ObjectQMgres otro gestor de colas del grupo de compartición de colas deben resolverse en una apertura de la cola compartida en el gestor de colas local. Este atributo es de sólo lectura.

suceso ssl
Indica si se generan sucesos SSL. Este atributo es de sólo lectura.

Se necesita SSL FIPS
Indica si sólo se deben utilizar algoritmos certificados por FIPS si la criptografía se ejecuta en el software de IBM MQ . Este atributo es de sólo lectura.

Cuenta restablecimiento clave SSL
Número de bytes no cifrados enviados y recibidos en una conversación SSL antes de que se renegocie la clave secreta. Este atributo es de sólo lectura.

suceso de inicio-detención
Controla los sucesos de inicio-detención. Este atributo es de sólo lectura.

Intervalo de estadísticas
Frecuencia con la que se graban los datos de supervisión de estadísticas en la cola de supervisión. Este atributo es de sólo lectura.

Disponibilidad de punto de sincronismo
Disponibilidad de participación de punto de sincronismo. Este atributo es de sólo lectura.

Nota: Las unidades de trabajo globales coordinadas por el gestor de colas no están soportadas en la plataforma IBM i . IBM i Puede programar una unidad de trabajo, coordinada externamente por IBM i, utilizando las llamadas al sistema nativo _Rcommit y _Rback. Inicie este tipo de unidad de trabajo iniciando la aplicación IBM MQ bajo control de compromiso a nivel de trabajo utilizando el mandato STRCMTCTL. Consulte Interfaces con el gestor de puntos de sincronismo externo de IBM i para obtener más detalles. La restitución y la confirmación están soportadas en la plataforma IBM i para unidades de trabajo locales coordinadas por un gestor de colas.

canales tcp
Número máximo de canales que pueden ser actuales o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP. Este atributo es de sólo lectura.

Mantener activo de TCP
Indica si se va a utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Este atributo es de sólo lectura.

Nombre TCP
El nombre del sistema TCP/IP único o predeterminado que se va a utilizar, en función del valor del tipo de pila tcp. Este atributo es de sólo lectura.

Tipo de pila TCP
Indica si el iniciador de canal sólo puede utilizar el espacio de direcciones TCP/IP especificado en el nombre tcp o puede enlazarse a cualquier dirección TCP/IP seleccionada. Este atributo es de sólo lectura.

Registro de la ruta de rastreo
Controla el registro de la información de rastreo de ruta. Este atributo es de sólo lectura.

Activar intervalo
Intervalo de desencadenante. Este atributo es de sólo lectura.

ID de usuario
En plataformas UNIX and Linux , el ID de usuario real de la aplicación. En plataformas Windows , el ID de usuario de la aplicación.

Constructores

ImqQueueManager ();
El constructor predeterminado.

ImqQueueManager (const ImqQueueGestor & *gestor*);
El constructor de copia. El estado de conexión será FALSE.

ImqQueueManager (const char * *nombre*);
Establece el nombre de ImqObject en *name*.

Destructores

Cuando se destruye un objeto del gestor ImqQueue, se desconecta automáticamente.

Métodos de clase (public)

comportamiento MQLONG estático ();
Devuelve el comportamiento.

void setBehavior(const MQLONG *behavior* = 0);
Establece el comportamiento.

Métodos de objeto (public)

void operator = (const ImqQueueGestor & *mgr*);
Se desconecta si es necesario y copia los datos de instancia de *mgr*. El estado de conexión es FALSE.

ImqBoolean accountingConnOverride (MQLONG & statint);
Proporciona una copia del valor de alteración temporal de conexiones de contabilidad. Devuelve TRUE si es satisfactorio.

MQLONG accountingConnOverride ();
Devuelve el valor de alteración temporal de conexiones de contabilidad sin ninguna indicación de posibles errores.

ImqBoolean accountingInterval (MQLONG & statint);
Proporciona una copia del valor de intervalo de contabilidad. Devuelve TRUE si es satisfactorio.

MQLONG accountingInterval ();
Devuelve el valor de intervalo de contabilidad sin ninguna indicación de posibles errores.

ImqBoolean activityRecording (MQLONG & rec);
Proporciona una copia del valor de registro de actividad. Devuelve TRUE si es satisfactorio.

MQLONG activityRecording ();
Devuelve el valor de registro de actividad sin ninguna indicación de posibles errores.

ImqBoolean adoptNewMCACheck (MQLONG & check);
Proporciona una copia del valor de comprobación de adopción de MCA nuevo. Devuelve TRUE si es satisfactorio.

MQLONG adoptNewMCACheck ();
Devuelve el valor de comprobación de adopción de nuevo MCA sin ninguna indicación de posibles errores.

ImqBoolean adoptNewMCAType (MQLONG & type);
Proporciona una copia del nuevo tipo de MCA de adopción. Devuelve TRUE si es satisfactorio.

MQLONG adoptNewMCAType ();
Devuelve el nuevo tipo de MCA de adopción sin ninguna indicación de posibles errores.

QLONG authenticationType () Const.
Devuelve el tipo de autenticación.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);
Establece el tipo de autenticación.

ImqBoolean authorityEvent(suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso de autorización. Devuelve TRUE si es satisfactorio.

MQLONG authorityEvent();
Devuelve el estado de habilitación del suceso de autorización sin ninguna indicación de posibles errores.

ImqBoolean backout ();
Restituye los cambios no confirmados. Devuelve TRUE si es satisfactorio.

ImqBoolean begin ();
Inicia una unidad de trabajo. Las opciones de inicio afectan al comportamiento de este método. Devuelve TRUE si es satisfactorio, pero también devuelve TRUE incluso si la llamada MQBEGIN subyacente devuelve MQRC_NO_EXTERNAL_PARTICIPANTES o MQRC_PARTICIPANT_NOT_AVAILABLE (que están asociados con MQCC_WARNING).

MQLONG beginOptions() Const.
Devuelve las opciones de inicio.

void setBeginOptions (const MQLONG *options* = MQBO_NONE);
Establece las opciones de inicio.

ImqBoolean bridgeEvent (MQLONG & event);
Proporciona una copia del valor de suceso de puente. Devuelve TRUE si es satisfactorio.

MQLONG bridgeEvent ();
Devuelve el valor de suceso de puente sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinition (MQLONG & *valor*);
Proporciona una copia del valor de definición automática de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelAutoDefinition ();
Devuelve el valor de definición automática de canal sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinitionEvent(MQLONG & *valor*);
Proporciona una copia del valor de suceso de definición automática de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelAutoDefinitionEvent();
Devuelve el valor de suceso de definición automática de canal sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinitionExit(ImqString & *nombre*);
Proporciona una copia del nombre de salida de definición automática de canal. Devuelve TRUE si es satisfactorio.

ImqString channelAutoDefinitionExit( );
Devuelve el nombre de salida de definición automática de canal sin ninguna indicación de posibles errores.

ImqBoolean channelEvent (MQLONG & event);
Proporciona una copia del valor de suceso de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelEvent();
Devuelve el valor de suceso de canal sin ninguna indicación de posibles errores.

MQLONG channelInitiatorAdapters ();
Devuelve el valor de adaptadores de iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorAdapters (MQLONG & adapters);
Proporciona una copia del valor de adaptadores de iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorControl ();
Devuelve el valor de inicio del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorControl (MQLONG & init);
Proporciona una copia del valor de inicio de control del iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorDispatchers ();
Devuelve el valor de los asignadores de iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorDispatchers (MQLONG & dispatchers);
Proporciona una copia del valor de los asignadores del iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorTraceAutoInicio ();
Devuelve el valor de inicio automático del rastreo del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorTraceAutoInicio (MQLONG & auto);
Proporciona una copia del valor de inicio automático de rastreo de iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorTraceTableTamaño ();
Devuelve el valor de tamaño de tabla de rastreo del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorTraceTableTamaño (MQLONG & size);
Proporciona una copia del valor de tamaño de tabla de rastreo del iniciador de canal. Devuelve TRUE si es satisfactorio.

ImqBoolean channelMonitoring (MQLONG & monchl);
Proporciona una copia del valor de supervisión de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelMonitoring ();
Devuelve el valor de supervisión de canal sin ninguna indicación de posibles errores.

ImqBoolean channelReference(ImqChannel * & *pchannel*);
Proporciona una copia de la referencia de canal. Si la referencia de canal no es válida, establece *pchannel* en nulo. Este método devuelve TRUE si es satisfactorio.

ImqChannel * channelReference( );
Devuelve la referencia de canal sin ninguna indicación de posibles errores.

ImqBoolean setChannelReference (ImqChannel & *canal*);
Establece la referencia de canal. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setChannelReference (ImqChannel * *channel* = 0);
Establece o restablece la referencia de canal. Este método devuelve TRUE si es satisfactorio.

ImqBoolean channelStatistics (MQLONG & statchl);
Proporciona una copia del valor de estadísticas de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelStatistics ();
Devuelve el valor de estadísticas de canal sin ninguna indicación de posibles errores.

ImqBoolean characterSet(MQLONG & *ccsid*);
Proporciona una copia del juego de caracteres. Devuelve TRUE si es satisfactorio.

MQLONG characterSet();
Devuelve una copia del juego de caracteres, sin ninguna indicación de posibles errores.

MQLONG clientSslKeyResetCount () Const.
Devuelve el valor de recuento de restablecimiento de clave SSL utilizado en las conexiones de cliente.

void setClientSslKeyResetCount(const MQLONG count);
Establece el recuento de restablecimiento de clave SSL utilizado en las conexiones de cliente.

ImqBoolean clusterSenderMonitoring (MQLONG & monacls);
Proporciona una copia del valor predeterminado de supervisión del remitente del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterSenderMonitoring ();
Devuelve el valor predeterminado de supervisión del remitente del clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterSenderStatistics (MQLONG & statacls);
Proporciona una copia del valor de estadísticas del remitente del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterSenderStatistics ();
Devuelve el valor de estadísticas de clúster emisor sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadDatos (ImqString & *datos*);
Proporciona una copia de los datos de salida de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

ImqString clusterWorkloadData ();
Devuelve los datos de salida de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadExit (ImqString & *nombre*);
Proporciona una copia del nombre de salida de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

ImqString clusterWorkloadExit ();
Devuelve el nombre de salida de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadLongitud (MQLONG & *longitud*);
Proporciona una copia de la longitud de carga de trabajo del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkloadLongitud ();
Devuelve la longitud de carga de trabajo del clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
Proporciona una copia del valor de canales utilizados más recientemente de la carga de trabajo del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadMRU ();
Devuelve el valor de canales utilizados más recientemente de la carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Proporciona una copia del valor de cola de uso de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadUseQ ();
Devuelve el valor de cola de uso de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean commandEvent (MQLONG & event);
Proporciona una copia del valor de suceso de mandato. Devuelve TRUE si es satisfactorio.

MQLONG commandEvent ();
Devuelve el valor de suceso de mandato sin ninguna indicación de posibles errores.

ImqBoolean commandInputQueueName(ImqString & *nombre*);
Proporciona una copia del nombre de cola de entrada de mandatos. Devuelve TRUE si es satisfactorio.

ImqString commandInputQueueName( );
Devuelve el nombre de cola de entrada de mandatos sin ninguna indicación de posibles errores.

ImqBoolean commandLevel(MQLONG & *nivel*);
Proporciona una copia del nivel de mandatos. Devuelve TRUE si es satisfactorio.

MQLONG commandLevel();
Devuelve el nivel de mandatos sin ninguna indicación de posibles errores.

MQLONG commandServerControl ();
Devuelve el valor de inicio del servidor de mandatos sin ninguna indicación de posibles errores.

ImqBoolean commandServerControl (MQLONG & server);
Proporciona una copia del valor de inicio de control del servidor de mandatos. Devuelve TRUE si es satisfactorio.

ImqBoolean commit ();
Confirma los cambios no confirmados. Devuelve TRUE si es satisfactorio.

ImqBoolean connect ();
Se conecta al gestor de colas con el nombre ImqObject especificado, siendo el valor predeterminado el gestor de colas local. Si desea conectarse a un gestor de colas específico, utilice el método ImqObject setName antes de la conexión. Si hay una referencia de canal, se utiliza para pasar información sobre la definición de canal a MQCONNX en un MQCD. El ChannelType en MQCD se establece en MQCHT_CLNTCONN. La información de referencia de canal, que sólo es significativa para las conexiones de cliente, se ignora para las conexiones de servidor. Las opciones de conexión afectan al comportamiento de este método. Este método establece el estado de conexión en TRUE si es satisfactorio. Devuelve el nuevo estado de conexión.

Si hay un primer registro de autenticación, la cadena de registros de autenticación se utiliza para autenticar certificados digitales para canales de cliente seguros.

Puede conectar más de un objeto de gestor ImqQueueal mismo gestor de colas. Todos utilizan el mismo descriptor de conexión MQHCONN y comparten la funcionalidad de UOW para la conexión

asociada con la hebra. El primer gestor ImqQueueque se conecta obtiene el descriptor de contexto MQHCONN. El último gestor de ImqQueueen desconectarse realiza el MQDISC.

Para un programa multihebra, se recomienda que se utilice un objeto de gestor ImqQueueindependiente para cada hebra.

ImqBinary connectionId () Const.
Devuelve el ID de conexión.

ImqBinary connectionTag () Const.
Devuelve la etiqueta de conexión.

ImqBoolean setConnectionTag (const MQBYTE128 *tag* = 0);
Establece la etiqueta de conexión. Si *tag* es cero, borra la etiqueta de conexión. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setConnectionTag (const ImqBinary & *etiqueta*);
Establece la etiqueta de conexión. La longitud de datos del *código* debe ser cero (para borrar el código de conexión) o MQ_CONN_TAG_LENGTH. Este método devuelve TRUE si es satisfactorio.

MQLONG connectOptions() Const.
Devuelve las opciones de conexión.

void setConnectOptions (const MQLONG *options* = MQCNO_NONE);
Establece las opciones de conexión.

ImqBoolean connectionStatus() Const.
Devuelve el estado de conexión.

ImqString cryptographicHardware ( );
Devuelve el hardware criptográfico.

ImqBoolean setCryptographicHardware (const char * *hardware* = 0);
Establece el hardware criptográfico. Este método devuelve TRUE si es satisfactorio.

ImqBoolean deadLetterQueueName(ImqString & *nombre*);
Proporciona una copia del nombre de cola de mensajes no entregados. Devuelve TRUE si es satisfactorio.

ImqString deadLetterQueueName( );
Devuelve una copia del nombre de cola de mensajes no entregados, sin ninguna indicación de posibles errores.

ImqBoolean defaultTransmissionQueueName(ImqString & *nombre*);
Proporciona una copia del nombre de cola de transmisión predeterminado. Devuelve TRUE si es satisfactorio.

ImqString defaultTransmissionQueueName( );
Devuelve el nombre de cola de transmisión predeterminado sin ninguna indicación de posibles errores.

ImqBoolean disconnect ();
Se desconecta del gestor de colas y establece el estado de conexión en FALSE. Cierra todos los objetos ImqProcess e ImqQueue asociados con este objeto, y interrumpe su referencia de conexión antes de la desconexión. Si hay más de un objeto de gestor ImqQueueconectado al mismo gestor de colas, sólo el último en desconectarse realiza una desconexión física; otros realizan una desconexión lógica. Los cambios no confirmados sólo se confirman en la desconexión física.

Este método devuelve TRUE si es satisfactorio. Si se llama cuando no hay ninguna conexión existente, el código de retorno también es verdadero.

ImqBoolean distributionLists(soporte de MQLONG &);
Proporciona una copia del valor de las listas de distribución. Devuelve TRUE si es satisfactorio.

MQLONG distributionLists();
Devuelve el valor de las listas de distribución sin ninguna indicación de posibles errores.

ImqBoolean dnsGroup (ImqString & group);
Proporciona una copia del nombre de grupo DNS. Devuelve TRUE si es satisfactorio.

ImqString dnsGroup ( );
Devuelve el nombre del grupo DNS sin ninguna indicación de posibles errores.

ImqBoolean dnsWlm (MQLONG & wlm);
Proporciona una copia del valor WLM de DNS. Devuelve TRUE si es satisfactorio.

MQLONG dnsWlm ();
Devuelve el valor WLM de DNS sin ninguna indicación de posibles errores.

ImqAuthenticationRecord * firstAuthenticationRecord () Const.
Devuelve el primer registro de autenticación.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * *air* = 0);
Establece el primer registro de autenticación.

ImqObject * firstManagedObject () Const.
Devuelve el primer objeto gestionado.

ImqBoolean inhibitEvent(suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso de inhibición. Devuelve TRUE si es satisfactorio.

MQLONG inhibitEvent();
Devuelve el estado de habilitación del suceso de inhibición sin ninguna indicación de posibles errores.

ImqBoolean ipAddressVersion (MQLONG & version);
Proporciona una copia del valor de versión de dirección IP. Devuelve TRUE si es satisfactorio.

MQLONG ipAddressVersión ();
Devuelve el valor de versión de dirección IP sin ninguna indicación de posibles errores.

ImqBoolean keepAlive (MQLONG & keepalive);
Proporciona una copia del valor de mantener activo. Devuelve TRUE si es satisfactorio.

MQLONG keepAlive ();
Devuelve el valor de mantener activo sin ninguna indicación de posibles errores.

ImqString keyRepository ( );
Devuelve el repositorio de claves.

ImqBoolean setKeyRepository (const char * *repositorio* = 0);
Establece el repositorio de claves. Devuelve TRUE si es satisfactorio.

ImqBoolean listenerTimer (MQLONG & timer);
Proporciona una copia del valor de temporizador de escucha. Devuelve TRUE si es satisfactorio.

MQLONG listenerTimer ();
Devuelve el valor de temporizador de escucha sin ninguna indicación de posibles errores.

ImqBoolean localEvent(suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso local. Devuelve TRUE si es satisfactorio.

MQLONG localEvent();
Devuelve el estado de habilitación del suceso local sin ninguna indicación de posibles errores.

ImqBoolean loggerEvent (MQLONG & count);
Proporciona una copia del valor de suceso de registrador. Devuelve TRUE si es satisfactorio.

MQLONG loggerEvent ();
Devuelve el valor de suceso de registrador sin ninguna indicación de posibles errores.

ImqBoolean luGroupNombre (ImqString & name);.
Proporciona una copia del nombre de grupo de LU. Devuelve TRUE si es satisfactorio

ImqString luGroupNombre ();
Devuelve el nombre del grupo de LU sin ninguna indicación de posibles errores.

ImqBoolean lu62ARMSuffix (ImqString & suffix);
Proporciona una copia del sufijo ARM LU62 . Devuelve TRUE si es satisfactorio.

ImqString lu62ARMSuffix ( );
Devuelve el sufijo ARM LU62 sin ninguna indicación de posibles errores

ImqBoolean luName (ImqString & name);
Proporciona una copia del nombre de LU. Devuelve TRUE si es satisfactorio.

ImqString luName ( );
Devuelve el nombre de LU sin ninguna indicación de posibles errores.

ImqBoolean maximumActiveChannels (MQLONG & canales);
Proporciona una copia del valor máximo de canales activos. Devuelve TRUE si es satisfactorio.

MQLONG maximumActiveChannels ();
Devuelve el valor máximo de canales activos sin ninguna indicación de posibles errores.

ImqBoolean maximumCurrentChannels (MQLONG & channels);
Proporciona una copia del valor máximo de canales actuales. Devuelve TRUE si es satisfactorio.

MQLONG maximumCurrentChannels ();
Devuelve el valor máximo de canales actuales sin ninguna indicación de posibles errores.

ImqBoolean maximumHandles(MQLONG & *número*);
Proporciona una copia de los manejadores máximos. Devuelve TRUE si es satisfactorio.

MQLONG maximumHandles();
Devuelve los descriptores de contexto máximos sin ninguna indicación de posibles errores.

ImqBoolean maximumLu62Channels (MQLONG & canales);
Proporciona una copia del valor máximo de canales LU62 . Devuelve TRUE si es satisfactorio.

MQLONG maximumLu62Channels ();.
Devuelve el valor máximo de canales LU62 sin ninguna indicación de posibles errores

ImqBoolean maximumMessageLength (MQLONG & *longitud*);
Proporciona una copia de la longitud máxima del mensaje. Devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLength ();
Devuelve la longitud máxima del mensaje sin ninguna indicación de posibles errores.

ImqBoolean maximumPriority(MQLONG & *prioridad*);
Proporciona una copia de la prioridad máxima. Devuelve TRUE si es satisfactorio.

MQLONG maximumPriority();
Devuelve una copia de la prioridad máxima, sin ninguna indicación de posibles errores.

ImqBoolean maximumTcpCanales (MQLONG & canales);
Proporciona una copia del valor máximo de canales TCP. Devuelve TRUE si es satisfactorio.

MQLONG maximumTcpCanales ();
Devuelve el valor máximo de canales TCP sin ninguna indicación de posibles errores.

ImqBoolean maximumUncommittedMessages (MQLONG & *número*);
Proporciona una copia del número máximo de mensajes no confirmados. Devuelve TRUE si es satisfactorio.

MQLONG maximumUncommittedMensajes ();
Devuelve el número máximo de mensajes no confirmados sin ninguna indicación de posibles errores.

ImqBoolean mqiAccounting (MQLONG & statint);
Proporciona una copia del valor de contabilidad MQI. Devuelve TRUE si es satisfactorio.

MQLONG mqiAccounting ();
Devuelve el valor de contabilidad MQI sin ninguna indicación de posibles errores.

ImqBoolean mqiStatistics (MQLONG & statmqi);
Proporciona una copia del valor de estadísticas de MQI. Devuelve TRUE si es satisfactorio.

MQLONG mqiStatistics ();
Devuelve el valor de estadísticas MQI sin ninguna indicación de posibles errores.

ImqBoolean outboundPortMax (MQLONG & max);
Proporciona una copia del valor máximo de puerto de salida. Devuelve TRUE si es satisfactorio.

MQLONG outboundPortMax ();
Devuelve el valor de puerto de salida máximo sin ninguna indicación de posibles errores.

ImqBoolean outboundPortMín (MQLONG & min);
Proporciona una copia del valor de puerto de salida mínimo. Devuelve TRUE si es satisfactorio.

MQLONG outboundPortMin ();
Devuelve el valor de puerto de salida mínimo sin ninguna indicación de posibles errores.

Contraseña de ImqBinary () Const.
Devuelve la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const ImqString & password);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const char * = 0 contraseña);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const ImqBinary & password);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean performanceEvent(suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso de rendimiento. Devuelve TRUE si es satisfactorio.

MQLONG performanceEvent();
Devuelve el estado de habilitación del suceso de rendimiento sin ninguna indicación de posibles errores.

Plataforma ImqBoolean (plataforma MQLONG &);
Proporciona una copia de la plataforma. Devuelve TRUE si es satisfactorio.

plataforma MQLONG ();
Devuelve la plataforma sin ninguna indicación de posibles errores.

ImqBoolean queueAccounting (MQLONG & acctq);
Proporciona una copia del valor de contabilidad de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueAccounting ();
Devuelve el valor de contabilidad de cola sin ninguna indicación de posibles errores.

ImqBoolean queueMonitoring (MQLONG & monq);
Proporciona una copia del valor de supervisión de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueMonitoring ();
Devuelve el valor de supervisión de cola sin ninguna indicación de posibles errores.

ImqBoolean queueStatistics (MQLONG & statq);
Proporciona una copia del valor de estadísticas de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueStatistics ();
Devuelve el valor de estadísticas de cola sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeout (MQLONG & timeout);
Proporciona una copia del valor de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeout ();
Devuelve el valor de tiempo de espera de recepción sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeoutMin (MQLONG & min);
Proporciona una copia del valor mínimo de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeoutMin ();
Devuelve el valor de tiempo de espera de recepción mínimo sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeoutTipo (MQLONG & type);
Proporciona una copia del tipo de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeoutTipo ();
Devuelve el tipo de tiempo de espera de recepción sin ninguna indicación de posibles errores.

ImqBoolean remoteEvent(suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso remoto. Devuelve TRUE si es satisfactorio.

MQLONG remoteEvent();
Devuelve el estado de habilitación del suceso remoto sin ninguna indicación de posibles errores.

ImqBoolean repositoryName(ImqString & *nombre*);
Proporciona una copia del nombre de repositorio. Devuelve TRUE si es satisfactorio.

ImqString repositoryName( );
Devuelve el nombre del repositorio sin ninguna indicación de posibles errores.

ImqBoolean repositoryNamelistNombre (ImqString & *nombre*);
Proporciona una copia del nombre de la lista de nombres del repositorio. Devuelve TRUE si es satisfactorio.

ImqString repositoryNamelistNombre ();
Devuelve una copia del nombre de la lista de nombres del repositorio sin ninguna indicación de posibles errores.

ImqBoolean sharedQueueQueueManagerNombre (MQLONG & name);
Proporciona una copia del valor del nombre del gestor de colas compartido. Devuelve TRUE si es satisfactorio.

MQLONG sharedQueueQueueManagerNombre ();
Devuelve el valor del nombre del gestor de colas compartido sin ninguna indicación de posibles errores.

ImqBoolean sslEvent (MQLONG & event);
Proporciona una copia del valor de suceso SSL. Devuelve TRUE si es satisfactorio.

MQLONG sslEvent ();
Devuelve el valor de suceso SSL sin ninguna indicación de posibles errores.

ImqBoolean sslFips (MQLONG & sslfips);
Proporciona una copia del valor SSL FIPS. Devuelve TRUE si es satisfactorio.

MQLONG sslFips ();
Devuelve el valor SSL FIPS sin ninguna indicación de posibles errores.

ImqBoolean sslKeyResetCount (MQLONG & count);
Proporciona una copia del valor de recuento de restablecimiento de clave SSL. Devuelve TRUE si es satisfactorio.

MQLONG sslKeyResetCount ();
Devuelve el valor de recuento de restablecimiento de clave SSL sin ninguna indicación de posibles errores.

ImqBoolean startStopSuceso (suceso MQLONG &);
Proporciona una copia del estado de habilitación del suceso de inicio-detención. Devuelve TRUE si es satisfactorio.

MQLONG startStopSuceso ();
Devuelve el estado de habilitación del suceso de inicio-detención sin ninguna indicación de posibles errores.

ImqBoolean statisticsInterval (MQLONG & statint);
Proporciona una copia del valor de intervalo de estadísticas. Devuelve TRUE si es satisfactorio.

MQLONG statisticsInterval ();
Devuelve el valor de intervalo de estadísticas sin ninguna indicación de posibles errores.

ImqBoolean syncPointAvailability (MQLONG & *sincronización*);
Proporciona una copia del valor de disponibilidad de punto de sincronismo. Devuelve TRUE si es satisfactorio.

MQLONG syncPointDisponibilidad ();
Devuelve una copia del valor de disponibilidad de punto de sincronismo, sin ninguna indicación de posibles errores.

ImqBoolean tcpName (ImqString & name);
Proporciona una copia del nombre del sistema TCP. Devuelve TRUE si es satisfactorio.

ImqString tcpName ( );
Devuelve el nombre del sistema TCP sin ninguna indicación de posibles errores.

ImqBoolean tcpStackTipo (MQLONG & type);
Proporciona una copia del tipo de pila TCP. Devuelve TRUE si es satisfactorio.

MQLONG tcpStackTipo ();
Devuelve el tipo de pila TCP sin ninguna indicación de posibles errores.

ImqBoolean traceRouteRecording (MQLONG & routerec);
Proporciona una copia del valor de registro de ruta de rastreo. Devuelve TRUE si es satisfactorio.

MQLONG traceRouteRecording ();
Devuelve el valor de registro de ruta de rastreo sin ninguna indicación de posibles errores.

ImqBoolean triggerInterval(MQLONG & *intervalo*);
Proporciona una copia del intervalo de desencadenante. Devuelve TRUE si es satisfactorio.

MQLONG triggerInterval();
Devuelve el intervalo de desencadenante sin ninguna indicación de posibles errores.

ImqBinary userId () Const.
Devuelve el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const ImqString & id);
Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const char * = 0 id);
Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const ImqBinary & id);
Establece el ID de usuario utilizado en las conexiones de cliente.

Métodos de objeto (protegidos)

void setFirstManagedObject (const ImqObject * *object* = 0);
Establece el primer objeto gestionado.

Datos de objeto (protegidos)

MQHCONN *ohconn*
El descriptor de conexión de IBM MQ (sólo es significativo mientras el estado de conexión es TRUE).

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- ERROR_ENTORNO_MQRC
- MQRC_FUNCTION_NOT_SUPPORTED
- ERROR_REFERENCIA_MQRC
- (códigos de razón para MQBACK)
- (códigos de razón para MQBEGIN)
- (códigos de razón para MQCMIT)
- (códigos de razón para MQCONNX)
- (códigos de razón para MQDISC)
- (códigos de razón para MQCONN)

Clase C++ de cabecera ImqReference

Esta clase encapsula características de la estructura de datos MQRMH.

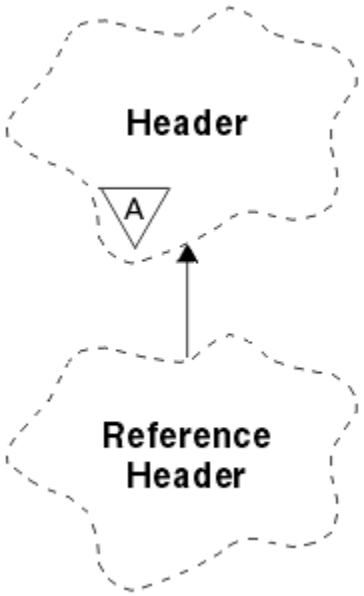


Figura 34. Clase de cabecera ImqReference

Esta clase está relacionada con las llamadas MQI listadas en “ImqReferenceReferencia cruzada de cabecera” en la página 1851.

- “Atributos de objetos” en la página 1934
- “Constructores” en la página 1935
- “Métodos ImqItem sobrecargados” en la página 1935
- “Métodos de objeto (public)” en la página 1935
- “Datos de objeto (protegidos)” en la página 1936
- “códigos de razón” en la página 1936

Atributos de objetos

entorno de destino
Entorno para el destino. El valor inicial es una serie nula.

nombre de destino
Nombre del destino de datos. El valor inicial es una serie nula.

ID de instancia
Identificador de instancia. Un valor binario (MQBYTE24) de longitud MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial es MQOII_NONE.

longitud lógica
Longitud lógica, o prevista, de los datos de mensaje que siguen a esta cabecera. El valor inicial es cero.

desplazamiento lógico
Desplazamiento lógico para los datos de mensaje siguientes, que se interpretarán en el contexto de los datos en su conjunto, en el destino final. El valor inicial es cero.

desplazamiento lógico 2
Extensión de orden superior al desplazamiento lógico. El valor inicial es cero.

Tipo de referencia
Tipo de referencia. El valor inicial es una serie nula.

Entorno de origen
Entorno para el origen. El valor inicial es una serie nula.

Nombre del origen
Nombre del origen de datos. El valor inicial es una serie nula.

Constructores

Cabecera ImqReference();
El constructor predeterminado.

ImqReferenceHeader (const ImqReferenceHeader & *header*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Inserta una estructura de datos MQRMH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en "Clase C++ ImqHeader" en la página 1879 para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Lee una estructura de datos MQRMH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato de ImqMessage debe ser MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en "Clase C++ ImqHeader" en la página 1879 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqReferenceHeader & *header*);
Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

ImqString destinationEnvironment () Const.
Devuelve una copia del entorno de destino.

void setDestinationEnvironment (const char * *entorno* = 0);
Establece el entorno de destino.

ImqString destinationName () Const.
Devuelve una copia del nombre de destino.

void setDestinationName (const char * *nombre* = 0);
Establece el nombre de destino.

ImqBinary instanceId () Const.
Devuelve una copia del ID de instancia.

ImqBoolean setInstanceId (const ImqBinary & *id*);
Establece el ID de instancia. La longitud de datos de *token* debe ser 0 o MQ_OBJECT_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setInstanceId (const MQBYTE24 *id* = 0);
Establece el ID de instancia. *id* puede ser cero, que es lo mismo que especificar MQOII_NONE. Si *id* es distinto de cero, debe direccionar MQ_OBJECT_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQOII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQOII_NONE.

MQLONG logicalLength () Const.
Devuelve la longitud lógica.

void setLogicalLength (const MQLONG *longitud*);
Establece la longitud lógica.

MQLONG logicalOffset () Const.
Devuelve el desplazamiento lógico.

void setLogicalOffset (const MQLONG *desplazamiento*);
Establece el desplazamiento lógico.

MQLONG logicalOffset2 () Const.
Devuelve el desplazamiento lógico 2.

void setLogicalOffset2 (const MQLONG *desplazamiento*);
Establece el desplazamiento lógico 2.

ImqString referenceType () Const.
Devuelve una copia del tipo de referencia.

void setReferenceType (const char * *nombre* = 0);
Establece el tipo de referencia.

ImqString sourceEnvironment () Const.
Devuelve una copia del entorno de origen.

void setSourceEnvironment (const char * *entorno* = 0);
Establece el entorno de origen.

ImqString sourceName () Const.
Devuelve una copia del nombre de origen.

void setSourceName (const char * *nombre* = 0);
Establece el nombre de origen.

Datos de objeto (protegidos)

MQRMH *omqrmh*
Estructura de datos MQRMH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INSUFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR

Clase C++ ImqString

Esta clase proporciona almacenamiento de serie de caracteres y manipulación para series terminadas en nulo.

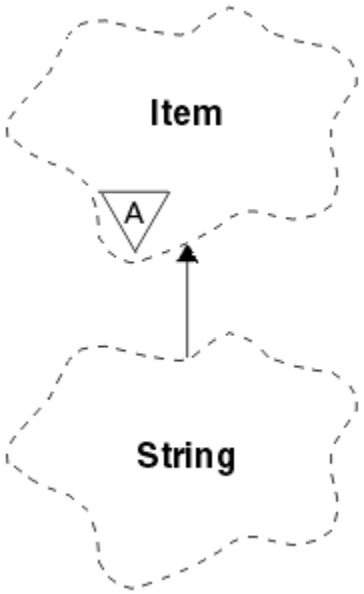


Figura 35. Clase ImqString

Utilice un ImqString en lugar de un **char *** en la mayoría de las situaciones en las que un parámetro llama a un **char ***.

- “Atributos de objetos” en la página 1937
- “Constructores” en la página 1937
- “Métodos de clase (public)” en la página 1938
- “Métodos ImqItem sobrecargados” en la página 1938
- “Métodos de objeto (public)” en la página 1938
- “Métodos de objeto (protegidos)” en la página 1941
- “códigos de razón” en la página 1941

Atributos de objetos

caracteres
Caracteres del **almacenamiento** que preceden a un nulo final.

length
Número de bytes en los **caracteres**. Si no hay ningún **almacenamiento**, la **longitud** es cero. El valor inicial es cero.

almacenamiento
Una matriz volátil de bytes de tamaño arbitrario. Un nulo final siempre debe estar presente en el **almacenamiento** después de los **caracteres**, para que se pueda detectar el final de los **caracteres** . Los métodos aseguran que esta situación se mantiene, pero aseguran, cuando se establecen los bytes en la matriz directamente, que existe un nulo final después de la modificación. Inicialmente, no hay ningún atributo **storage** .

Constructores

ImqString( );
El constructor predeterminado.

ImqString(const ImqString & *serie*);
El constructor de copia.

ImqString(const char *c*);
Los **caracteres** comprenden *c*.

ImqString(const char * *texto*);
Los **caracteres** se copian de *texto*.

ImqString(const void * *buffer*, const size_t *longitud*);
Copia *longitud* bytes a partir de *almacenamiento intermedio* y los asigna a los **caracteres**. La sustitución se realiza para cualquier carácter nulo copiado. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado.

Métodos de clase (public)

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *buffer-origen*, const char *pad* = 0);
Copia hasta *longitud* bytes de *source-buffer* a *destination-buffer*. Si el número de caracteres en *almacenamiento intermedio de origen* es insuficiente, rellena el espacio restante en *almacenamiento intermedio de destino* con *relleno* caracteres. *source-buffer* puede ser cero. *destination-buffer* puede ser cero si *length* también es cero. Los códigos de error se pierden. Este método devuelve TRUE si es satisfactorio.

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *buffer-origen*, ImqError & *objeto-error*, const char *pad* = 0);
Copia hasta *longitud* bytes de *source-buffer* a *destination-buffer*. Si el número de caracteres en *almacenamiento intermedio de origen* es insuficiente, rellena el espacio restante en *almacenamiento intermedio de destino* con *relleno* caracteres. *source-buffer* puede ser cero. *destination-buffer* puede ser cero si *length* también es cero. Los códigos de error se establecen en *objeto de error*. Este método devuelve TRUE si es satisfactorio.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Copia los **caracteres** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Establece los **caracteres** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **caracteres**existentes.

Para que la operación sea satisfactoria, la **codificación** del objeto *msg* debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

char & operator [] (const size_t *desplazamiento*) Const.
Hace referencia al carácter en desplazamiento *offset* en el **almacenamiento**. Asegúrese de que el byte relevante existe y es direccionable.

Operador ImqString () (const size_t *desplazamiento*, const size_t *longitud* = 1) Const.
Devuelve una subserie copiando bytes de los **caracteres** a partir de *desplazamiento*. Si *length* es cero, devuelve el resto de los **caracteres**. Si la combinación de *offset* y *length* no produce una referencia dentro de los **caracteres**, devuelve una ImqStringvacía.

void operator = (const ImqString & *serie*);
Copia datos de instancia de *serie*, sustituyendo los datos de instancia existentes.

Operador ImqString + (const char *c*) Const.
Devuelve el resultado de añadir *c* a los **caracteres**.

ImqString operator + (const char * *texto*) Const.
Devuelve el resultado de añadir *text* a los **caracteres**. Esto también se puede invertir. Por ejemplo:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Nota: Aunque la mayoría de compiladores aceptan **strOne + "string two";** Microsoft Visual C++ requiere **strOne + (char *) "string two";**

Operador ImqString + (const ImqString & *string1*) Const.
Devuelve el resultado de añadir *string1* a los **caracteres**.

ImqString operator + (const double *número*) Const.
Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

Operador ImqString + (const long *número*) Const.
Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

void operator + = (const char *c*);
Añade *c* a los **caracteres**.

void operator + = (const char * *texto*);
Añade *text* a los **caracteres**.

void operator + = (const ImqString & *serie*);
Añade *string* a los **caracteres**.

void operator + = (const double *número*);
Añade *número* a los **caracteres** después de la conversión a texto.

void operator + = (const long *número*);
Añade *número* a los **caracteres** después de la conversión a texto.

operador char * () Const.
Devuelve la dirección del primer byte en el **almacenamiento**. Este valor puede ser cero y es volátil. Utilice este método sólo para fines de sólo lectura.

Operador ImqBoolean < (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor que y FALSE si es mayor que o igual a.

Operador ImqBoolean > (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor que y FALSE si es menor o igual a.

Operador ImqBoolean < = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor o igual que y FALSE si es mayor que.

Operador ImqBoolean > = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor o igual que y FALSE si es menor que.

Operador ImqBoolean == (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

Operador ImqBoolean ! = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

comparación corta (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie*. El resultado es cero si los **caracteres** son iguales, negativo si es menor que y positivo si es mayor que. La comparación es sensible a las mayúsculas y minúsculas. Una ImqString nula se considera menor que una ImqStringno nula.

ImqBoolean copyOut(char * *buffer*, const size_t *length*, const char *pad* = 0);
Copia hasta *longitud* bytes de los **caracteres** en el *almacenamiento intermedio*. Si el número de **caracteres** es insuficiente, rellena el espacio restante en el *almacenamiento intermedio* con *relleno* caracteres. *buffer* puede ser cero si *length* también es cero. Devuelve TRUE si es satisfactorio.

size_t copyOut(long & *número*) Const.

Establece *número* de los **caracteres** después de la conversión a partir del texto y devuelve el número de caracteres implicados en la conversión. Si es cero, no se ha realizado ninguna conversión y no se ha establecido *número* . Una secuencia de caracteres convertible debe empezar con los valores siguientes:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & *señal*, const char *c* = '') Const.

Si los **caracteres** contienen uno o más caracteres que son diferentes de *c*, identifica una señal como la primera secuencia contigua de dichos caracteres. En este caso, *token* se establece en esa secuencia y el valor devuelto es la suma del número de caracteres iniciales *c* y el número de bytes de la secuencia. De lo contrario, devuelve cero y no establece *token*.

size_t cutOut(long & *número*);

Establece *número* como para el método **copy** , pero también elimina de **caracteres** el número de bytes indicado por el valor de retorno. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres números utilizando **cutOut** (*número*) tres veces:

```
strNumbers = "-1 0   +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & *señal*, const char *c* = " '

Establece *token* como para el método **copyOut** y elimina de **caracteres** los *strToken* caracteres y también los caracteres *c* que preceden a la *token* caracteres. Si *c* no está en blanco, elimina los caracteres *c* que se utilizan directamente para los caracteres *token* . Devuelve el número de caracteres eliminados. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres señales utilizando **cutOut** (*token*) tres veces:

```
strText = " Program Version 1.1 "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

El ejemplo siguiente muestra cómo analizar un nombre de vía de acceso de DOS:

```
strPath = "C:\OS2\BITMAP\OS2LOGO.BMP"

strPath.cutOut( strDrive, ':' );
strPath.stripLeading( ':' );
while ( strPath.cutOut( strFile, '\' ) );

// strDrive becomes "C".
// strFile becomes "OS2", then "BITMAP",
// then "OS2LOGO.BMP" leaving strPath empty.
```

ImqBoolean find (const ImqString & *serie*);

Busca una coincidencia exacta para *serie* en cualquier lugar dentro de los **caracteres**. Si no se encuentra ninguna coincidencia, devuelve FALSE. De lo contrario, devuelve TRUE. Si *string* es nulo, devuelve TRUE.

ImqBoolean find (const ImqString & *serie*, size_t & *desplazamiento*);

Busca una coincidencia exacta para *string* en algún lugar dentro de los **caracteres** del desplazamiento *offset* en adelante. Si *string* es nulo, devuelve TRUE sin actualizar *offset*. Si no se encuentra ninguna coincidencia, devuelve FALSE (es posible que se haya aumentado el valor de *offset*). Si se encuentra

una coincidencia, devuelve TRUE y actualiza *offset* al desplazamiento de *string* dentro de los **caracteres**.

size_t longitud () Const.
Devuelve la **longitud**.

ImqBoolean pasteIn(const double *número*, const char * *format* = "%f");
Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

La especificación *format* se utiliza para formatear la conversión de coma flotante. Si se especifica, debe ser uno adecuado para su uso con **printf** y números de coma flotante, por ejemplo **%.3f**.

ImqBoolean pasteIn(const long *número*);
Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

ImqBoolean pasteIn(const void * *buffer*, const size_t *longitud*);
Añade *longitud* bytes de *almacenamiento intermedio* a los **caracteres**y añade un valor nulo final. Sustituye los caracteres nulos copiados. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado. Este método devuelve TRUE si es satisfactorio.

ImqBoolean set (const char * *buffer*, const size_t *longitud*);
Establece los **caracteres** de un campo de caracteres de longitud fija, que puede contener un valor nulo. Añade un valor nulo a los caracteres del campo de longitud fija si es necesario. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setStorage(const size_t *longitud*);
Asigna (o reasigna) el **almacenamiento**. Conserva los **caracteres**originales, incluidos los nulos finales, si todavía hay espacio para ellos, pero no inicializa ningún almacenamiento adicional.

Este método devuelve TRUE si es satisfactorio.

size_t storage () Const.
Devuelve el número de bytes en el **almacenamiento**.

size_t stripLeading(const char *c* = '');
Elimina los caracteres iniciales *c* de los **caracteres** y devuelve el número eliminado.

size_t stripTrailing(const char *c* = '');
Elimina los caracteres de cola *c* de los **caracteres** y devuelve el número eliminado.

ImqString upperCase() Const.
Devuelve una copia en mayúsculas de los **caracteres**.

Métodos de objeto (protegidos)

ImqBoolean asignar (const ImqString & *serie*);
Equivalente al método **operator =** equivalente, pero no virtual. Devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_TRUNCADO
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT

Clase C++ ImqTrigger

Esta clase encapsula la estructura de datos MQTM (mensaje desencadenante).

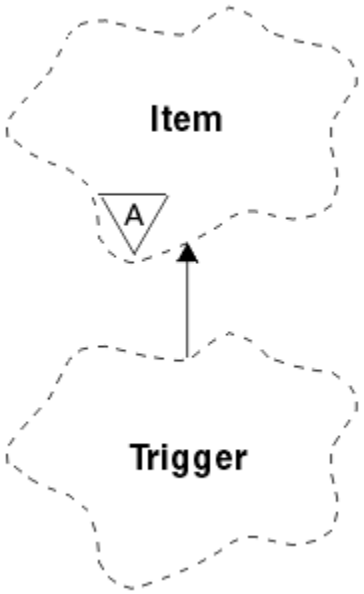


Figura 36. Clase ImqTrigger

Los objetos de esta clase suelen ser utilizados por un programa supervisor desencadenante. La tarea de un programa supervisor desencadenante es esperar estos mensajes concretos y actuar sobre ellos para asegurarse de que otras aplicaciones de IBM MQ se inician cuando los mensajes los esperan.

Consulte el programa de ejemplo IMQSTRG para ver un ejemplo de uso.

- “Atributos de objetos” en la página 1942
- “Constructores” en la página 1943
- “Métodos ImqItem sobrecargados” en la página 1943
- “Métodos de objeto (public)” en la página 1943
- “Datos de objeto (protegidos)” en la página 1944
- “códigos de razón” en la página 1944

Atributos de objetos

ID de aplicación
Identidad de la aplicación que ha enviado el mensaje. El valor inicial es una serie nula.

Tipo de aplicación
Tipo de aplicación que ha enviado el mensaje. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_DOS
- MQAT_IMS
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS

- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

Datos de entorno
Datos de entorno para el proceso. El valor inicial es una serie nula.

nombre de proceso
Nombre del proceso. El valor inicial es una serie nula.

nombre de cola
Nombre de la cola que se va a iniciar. El valor inicial es una serie nula.

Datos desencadenantes
Desencadenar datos para el proceso. El valor inicial es una serie nula.

datos de usuario
Datos de usuario para el proceso. El valor inicial es una serie nula.

Constructores

ImqTrigger( );
El constructor predeterminado.

ImqTrigger(const ImqTrigger & *desencadenante*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);
Escribe una estructura de datos MQTM en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en "Clase C++ ImqItem" en la página 1883 para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);
Lee una estructura de datos MQTM del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en "Clase C++ ImqItem" en la página 1883 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqTrigger & *desencadenante*);
Copia datos de instancia del *desencadenante*, sustituyendo los datos de instancia existentes.

ImqString applicationId () Const.
Devuelve una copia del ID de aplicación.

void setApplicationId (const char * *id*);
Establece el ID de aplicación.

MQLONG applicationType () Const.
Devuelve el tipo de aplicación.

void setApplicationType (const MQLONG *tipo*);
Establece el tipo de aplicación.

ImqBoolean copyOut (MQTMC2 * *ptmc2*);
Encapsula la estructura de datos MQTM, que es la que se recibe en las colas de inicio. Rellena una estructura de datos MQTMC2 equivalente proporcionada por el llamante y establece el campo QMgrName (que no está presente en la estructura de datos MQTM) en todos los espacios en blanco.

La estructura de datos MQTMC2 se utiliza tradicionalmente como parámetro para las aplicaciones iniciadas por un supervisor desencadenante. Este método devuelve TRUE si es satisfactorio.

ImqString environmentData () Const.
Devuelve una copia de los datos de entorno.

void setEnvironmentData (const char * *datos*);
Establece los datos de entorno.

ImqString processName () Const.
Devuelve una copia del nombre de proceso.

void setProcessName (const char * *nombre*);
Establece el nombre de proceso, rellenado con espacios en blanco en 48 caracteres.

ImqString queueName () Const.
Devuelve una copia del nombre de cola.

void setQueueName (const char * *nombre*);
Establece el nombre de cola, rellenando con espacios en blanco en 48 caracteres.

ImqString triggerData () Const.
Devuelve una copia de los datos de desencadenante.

void setTriggerData (const char * *datos*);
Establece los datos de desencadenante.

ImqString userData () Const.
Devuelve una copia de los datos de usuario.

void setUserData (const char * *datos*);
Establece los datos de usuario.

Datos de objeto (protegidos)

MQTM *omqtm*
Estructura de datos MQTM.

códigos de razón

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

Clase C++ de cabecera ImqWork

Esta clase encapsula características específicas de la estructura de datos MQWIH.

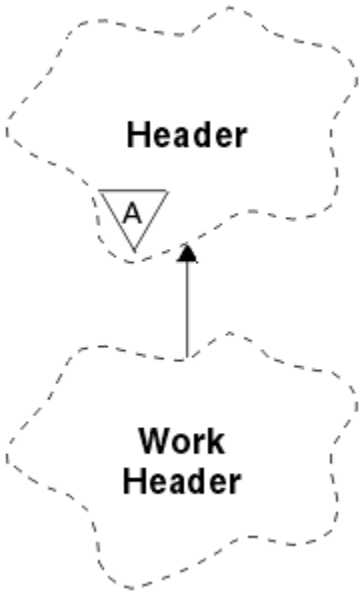


Figura 37. Clase de cabecera ImqWork

Los objetos de esta clase los utilizan las aplicaciones que colocan mensajes en la cola gestionada por el gestor de carga de trabajo de z/OS .

- "Atributos de objetos" en la página 1945
- "Constructores" en la página 1945
- "Métodos ImqItem sobrecargados" en la página 1945
- "Métodos de objeto (public)" en la página 1946
- "Datos de objeto (protegidos)" en la página 1946
- "códigos de razón" en la página 1946

Atributos de objetos

token de mensaje
Señal de mensaje para el gestor de carga de trabajo de z/OS , de longitud MQ_MSG_TOKEN_LENGTH. El valor inicial es MQMTOK_NONE.

nombre de servicio
Nombre de 32 caracteres de un proceso. El nombre está inicialmente en blanco.

paso de servicio
El nombre de 8 caracteres de un paso dentro del proceso. El nombre está inicialmente en blanco.

Constructores

Cabecera ImqWork();
El constructor predeterminado.

ImqWorkHeader (const ImqWorkHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQWIH al principio del almacenamiento intermedio de mensajes, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQWIH del almacenamiento intermedio de mensajes.

Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

El formato ImqMessage debe ser MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqWorkCabecera & *cabecera*);
Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

ImqBinary messageToken () Const.
Devuelve la **señal de mensaje**.

ImqBoolean setMessageToken (const ImqBinary & *token*);
Establece la **señal de mensaje**. La longitud de datos de *token* debe ser cero o MQ_MSG_TOKEN_LENGTH. Devuelve TRUE si es satisfactorio.

void setMessageToken (const MQBYTE16 *señal* = 0);
Establece la **señal de mensaje**. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe dirigirse a MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos como MQMTOK_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () Const.
Devuelve el **nombre de servicio**, incluidos los espacios en blanco finales.

void setServiceName (const char * *nombre*);
Establece el **nombre de servicio**.

ImqString serviceStep () Const.
Devuelve el **paso de servicio**, incluidos los espacios en blanco finales.

void setServiceStep (const char * *paso*);
Establece el **paso de servicio**.

Datos de objeto (protegidos)

MQWIH omqwih
Estructura de datos MQWIH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Propiedades de objetos IBM MQ classes for JMS

Todos los objetos de IBM MQ classes for JMS tienen propiedades. Las distintas propiedades se aplican a distintos tipos de objeto. Las distintas propiedades tienen distintos valores permitidos, y los valores de propiedades simbólicas difieren entre la herramienta de administración y el código de programa.

IBM MQ classes for JMS proporciona recursos para establecer y consultar las propiedades de los objetos utilizando la herramienta de administración de IBM MQ JMS , IBM MQ Explorer, o en una aplicación. Muchas de las propiedades sólo son relevantes para un subconjunto específico de los tipos de objeto.

Para obtener información sobre cómo utilizar la herramienta de administración de IBM MQ JMS , consulte Configuración de objetos de JMS utilizando la herramienta de administración.

Tabla 868 en la página 1947 proporciona una breve descripción de cada propiedad y muestra para cada propiedad a qué tipos de objeto se aplica. Los tipos de objeto se identifican utilizando palabras clave;

consulte Configuración de objetos de JMS utilizando la herramienta de administración para obtener una explicación de estos objetos.

Los números se refieren a las notas al final de la tabla. Consulte también el tema “Dependencias entre propiedades de objetos IBM MQ classes for JMS” en la página 1950.

Una propiedad consta de un par nombre-valor en el formato:

```
PROPERTY_NAME(property_value)
```

Los temas de esta sección listan, para cada propiedad, el nombre de la propiedad y una breve descripción, y muestra los valores de propiedad válidos utilizados en la herramienta de administración. y el método set que se utiliza para establecer el valor de la propiedad en una aplicación. Los temas también muestran los valores de propiedad válidos para cada propiedad y la correlación entre los valores de propiedad simbólica utilizados en la herramienta y sus equivalentes programables.

Los nombres de propiedad no distinguen entre mayúsculas y minúsculas y están restringidos al conjunto de nombres reconocidos que se muestran en estos temas.

Tabla 868. Nombres de propiedad y tipos de objeto aplicables

Propiedad	**Formato abreviado**	**Tipo de objeto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
“APPLICATIONNAME” en la página 1952	APPNAME	Y	Y	Y			Y	Y	Y
“ASYNCEXCEPTION” en la página 1953	AEX	Y	Y	Y			Y	Y	Y
“BROKERCCDURSUBQ” en la página 1954 [1]	CCDSUB					Y			
“BROKERCCSUBQ” en la página 1954 [1]	CCSUB	Y		Y			Y		Y
“BROKERCONQ” en la página 1954 [1]	BCON	Y		Y			Y		Y
“BROKERDURSUBQ” en la página 1955 [1]	BDSUB					Y			
“BROKERPUBQ” en la página 1955 [1]	BPUB	Y		Y		Y	Y		Y
“BROKERPUBQMGR” en la página 1956 [1]	BPQM					Y			
“BROKERQMGR” en la página 1956 [1]	BQM	Y		Y			Y		Y
“BROKERSUBQ” en la página 1957 [1]	BSUB	Y		Y			Y		Y
“BROKERVER” en la página 1957 [1]	BVER	Y [2]		Y [2]		Y	Y		Y
“CCDTURL” en la página 1958 [3]	CCDT	Y	Y	Y			Y	Y	Y
“CCSID” en la página 1958	CCS	Y	Y	Y	Y	Y	Y	Y	Y
“CHANNEL” en la página 1959 [3]	CHAN	Y	Y	Y			Y	Y	Y
“CLEANUP” en la página 1959 [1]	CL	Y		Y			Y		Y
“CLEANUPINT” en la página 1960 [1]	CLINT	Y		Y			Y		Y
“ConnectionNameList” en la página 1960	CNLISTA	Y	Y	Y					

Tabla 868. Nombres de propiedad y tipos de objeto aplicables (continuación)									
Propiedad	**Formato abreviado**	**Tipo de objeto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
“CLIENTRECONNECTOPTIONS” en la página 1961	CROPT	Y	Y	Y					
“CLIENTRECONNECTTIMEOUT” en la página 1962	CRT	Y	Y	Y					
“CLIENTID” en la página 1962	CID	Y [2]	Y	Y [2]			Y	Y	Y
“CLONESUPP” en la página 1963	CLS	Y		Y			Y		Y
“COMPHDR” en la página 1963	HC	Y		Y			Y		Y
“COMPMSG” en la página 1963	MC	Y	Y	Y			Y	Y	Y
“CONNOPT” en la página 1964	CNOPT	Y	Y	Y			Y	Y	Y
“CONNTAG” en la página 1965	CNTAG	Y	Y	Y			Y	Y	Y
“description” en la página 1965	DESC	Y [2]	Y	Y [2]	Y	Y	Y	Y	Y
“DIRECTAUTH” en la página 1966	DAUTH	Y [2]		Y [2]					
“ENCODING” en la página 1966	ENC				Y	Y			
“EXPIRY” en la página 1967	EXP				Y	Y			
“FAILIFQUIESCE” en la página 1968	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
“HOSTNAME” en la página 1969	HOST	Y [2]	Y	Y [2]			Y	Y	Y
“LOCALADDRESS” en la página 1969	LA	Y [2]	Y	Y [2]			Y	Y	Y
“ESTILO de nombre de mapa” en la página 1970	MNST	Y	Y	Y			Y	Y	Y
“MAXBUFFSIZE” en la página 1970	MBSZ	Y [2]		Y [2]					
“MDREAD” en la página 1971	MDR				Y	Y			
“MDWRITE” en la página 1971	MDW				Y	Y			
“MDMSGCTX” en la página 1972	MDCTX				Y	Y			
“MSGBATCHSZ” en la página 1973 [1]	MBS	Y	Y	Y			Y	Y	Y
“MSGBODY” en la página 1973	MBODY				Y	Y			
“MSGRETENTION” en la página 1974	MRET	Y	Y				Y	Y	
“MSGSELECTION” en la página 1974 [1]	MSEL	Y		Y			Y		Y
“MULTICAST” en la página 1975	MCAST	Y [2]		Y [2]		Y			
“OPTIMISTICPUBLICATION” en la página 1975 [1]	OPTPUB	Y		Y					
“OUTCOMENOTIFICATION” en la página 1976 [1]	NOTIFY	Y		Y					
“PERSISTENCE” en la página 1976	PER				Y	Y			
“POLLINGINT” en la página 1977 [1]	PINT	Y	Y	Y			Y	Y	Y
“PORT” en la página 1977	PORT	Y [2]	Y	Y [2]			Y	Y	Y

Tabla 868. Nombres de propiedad y tipos de objeto aplicables (continuación)									
Propiedad	**Formato abreviado**	**Tipo de objeto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
“PRIORITY” en la página 1978	PRI				Y	Y			
“PROCESSDURATION” en la página 1978 [1]	PROCDUR	Y		Y					
“PROVIDERVERSION” en la página 1979	PVER	Y	Y	Y			Y	Y	Y
“PROXYHOSTNAME” en la página 1981	PHOST	Y [2]		Y [2]					
“PROXYPORT” en la página 1982	PPORT	Y [2]		Y [2]					
“PUBACKINT” en la página 1982 [1]	PAI	Y		Y			Y		Y
“PUTASYNCALLOWED” en la página 1983	PAALD				Y	Y			
“QMANAGER” en la página 1983	QMGR	Y	Y	Y	Y		Y	Y	Y
“QUEUE” en la página 1984	QU				Y				
“READAHEADALLOWED” en la página 1984	RALD				Y	Y			
“READAHEADCLOSEPOLICY” en la página 1985	RACP				Y	Y			
“RECEIVECCSID” en la página 1985	RCCS				Y	Y			
“RECEIVECONVERSION” en la página 1986	RCNV				Y	Y			
“RECEIVEISOLATION” en la página 1986 [1]	RCVISOL	Y		Y					
“RECEXIT” en la página 1987	RCX	Y	Y	Y			Y	Y	Y
“RECEXITINIT” en la página 1987	RCXI	Y	Y	Y			Y	Y	Y
“REPLYTOSTYLE” en la página 1988	RTOST				Y	Y			
“RESCANINT” en la página 1988 [1]	RINT	Y	Y				Y	Y	
“SECEXIT” en la página 1989	SCX	Y	Y	Y			Y	Y	Y
“SECEXITINIT” en la página 1990	SCXI	Y	Y	Y			Y	Y	Y
“SENDCHECKCOUNT” en la página 1990	SCC	Y	Y	Y			Y	Y	Y
“SENDEXIT” en la página 1990	SDX	Y	Y	Y			Y	Y	Y
“SENDEXITINIT” en la página 1991	SDXI	Y	Y	Y			Y	Y	Y
“SHARECONVALLOWED” en la página 1991	SCALD	Y	Y	Y			Y	Y	Y
“SPARSESUBS” en la página 1992 [1]	SSUBS	Y		Y					
“SSLCIPHERSUITE” en la página 1993	SCPHS	Y	Y	Y			Y	Y	Y
“SSLCRL” en la página 1993	SCRL	Y	Y	Y			Y	Y	Y

Tabla 868. Nombres de propiedad y tipos de objeto aplicables (continuación)									
Propiedad	**Formato abreviado**	**Tipo de objeto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
“SSLFIPSREQUIRED” en la página 1994	SFIPS	Y	Y	Y			Y	Y	Y
“SSLPEERNAME” en la página 1994	SPEER	Y	Y	Y			Y	Y	Y
“SSLRESETCOUNT” en la página 1995	SRC	Y	Y	Y			Y	Y	Y
“STATREFRESHINT” en la página 1995 [1]	SRI	Y		Y			Y		Y
“SUBSTORE” en la página 1996 [1]	SS	Y		Y			Y		Y
“SYNCPOINTALLGETS” en la página 1996	SPAG	Y	Y	Y			Y	Y	Y
“TARGCLIENT” en la página 1997	TC				Y	Y			
“TARGCLIENTMATCHING” en la página 1997	TCM	Y	Y				Y	Y	
“TEMPMODEL” en la página 1998	TM	Y	Y				Y	Y	
“TEMPQPREFIX” en la página 1998	TQP	Y	Y				Y	Y	
“TEMPTOPICPREFIX” en la página 1999	TTP	Y		Y			Y		Y
“Tema” en la página 1999	SUPERIOR					Y			
“TRANSPORT” en la página 1999	TRAN	Y [2]	Y	Y [2]			Y	Y	Y
“WILDCARDFORMAT” en la página 2000	WCFMT	Y		Y			Y		Y

Nota:

1. Esta propiedad se puede utilizar con la versión 7.0 de IBM MQ classes for JMS pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad PROVIDERVERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
2. Solo se admiten las propiedades BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT y TRANSPORT para un objeto de fábrica ConnectionFactory o TopicConnectioncuando se utiliza una conexión en tiempo real con un intermediario.
3. Las propiedades CCDTURL y CHANNEL de un objeto no deben establecerse al mismo tiempo.

Dependencias entre propiedades de objetos IBM MQ classes for JMS

La validez de algunas propiedades depende de los valores particulares de otras propiedades.

Esta dependencia puede producirse en los siguientes grupos de propiedades:

- Propiedades del cliente
- Propiedades para una conexión en tiempo real con un intermediario
- Series de inicialización de salida

Propiedades del cliente

Para una conexión con un gestor de colas, las propiedades siguientes sólo son relevantes si TRANSPORT tiene el valor CLIENT:

- HOSTNAME
- PORT
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

No puede establecer valores para estas propiedades utilizando la herramienta de administración si TRANSPORT tiene el valor BIND.

Si TRANSPORT tiene el valor CLIENT, el valor predeterminado de la propiedad BROKERVER es V1 y el valor predeterminado de la propiedad PORT es 1414. Si establece el valor de BROKERVER o PORT explícitamente, un cambio posterior en el valor de TRANSPORT no altera temporalmente las opciones.

Propiedades para una conexión en tiempo real con un intermediario

Solo son relevantes las propiedades siguientes si TRANSPORT tiene el valor DIRECT o DIRECTHTTP:

- BROKERVER
- CLIENTID
- description
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (soportado solo para DIRECT)
- PORT
- PROXYHOSTNAME (soportado sólo para DIRECT)
- PROXYPORT (soportado sólo para DIRECT)

Si TRANSPORT tiene el valor DIRECT o DIRECTHTTP, el valor predeterminado de la propiedad BROKERVER es V2y el valor predeterminado de la propiedad PORT es 1506. Si establece el valor

de BROKERVER o PORT explícitamente, un cambio posterior en el valor de TRANSPORT no altera temporalmente las opciones.

Series de inicialización de salida
No establezca ninguna de las series de inicialización de salida sin proporcionar el nombre de salida correspondiente. Las propiedades de inicialización de salida son:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Por ejemplo, si especifica `RECEXITINIT(myString)` sin especificar `RECEXIT(some.exit.classname)` , se produce un error.

Referencia relacionada
"TRANSPORT" en la página 1999
La naturaleza de una conexión con un gestor de colas o intermediario.

APPLICATIONNAME

Una aplicación puede definir un nombre que identifique su conexión al gestor de colas. Este nombre de aplicación se muestra mediante el mandato **`DISPLAY CONN MQSC/PCF`** (donde el campo se denomina **`APPLTAG`**) o en la pantalla **Conexiones de aplicación** de IBM MQ Explorer (donde el campo se denomina **`App name`**).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : APPLICATIONNAME

Nombre abreviado de la herramienta de administración de JMS : APPNAME

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setAppNombre ()
- MQConnectionFactory.getAppNombre ()

Valores

Cualquier serie válida que no tenga más de 28 caracteres. Los nombres más largos se ajustan para ajustarse eliminando los nombres de paquete iniciales, si es necesario. Por ejemplo, si la clase invocadora es `com.example.MainApp`, se utiliza el nombre completo, pero si la clase invocadora es `com.example.dictionaryAndThesaurus.multilingual.mainApp`, se utiliza el nombre `multilingual.mainApp`, porque es la combinación más larga formada por el nombre de clase y el nombre de paquete situado más a la derecha que se ajusta a la longitud disponible.

Si el nombre de clase propiamente dicho tiene más de 28 caracteres de longitud, se trunca para que quepa. Por ejemplo, `com.example.mainApplicationForSecondTestCase` pasa a ser `mainApplicationForSecondTest`.

z/OS V 9.1.2 En z/OS, el APPNAME en:

- La modalidad de enlaces se ignora si se establece y, si se establece, sólo se puede establecer en espacios en blanco.
- La modalidad de cliente se puede establecer y utilizar.

ASYNCEXCEPTION

Esta propiedad determina si IBM MQ classes for JMS informa a un ExceptionListener sólo cuando se interrumpe una conexión o cuando se produce una excepción de forma asíncrona en una llamada de API de JMS . Esto se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : ASYNCEXCEPTION

Nombre abreviado de la herramienta de administración de JMS : AEX

Acceso programático

Métodos setter/Getters

- MQConnectionFactory.setAsyncExcepciones ()
- MQConnectionFactory.getAsyncExcepciones ()

Valores

ASÍNCRONOS_EXCEPCIONES_TODOS

Cualquier excepción detectada de forma asíncrona, fuera del ámbito de una llamada a API síncrona, y todas las excepciones de interrupción de conexión se envían al ExceptionListener.

Tabla 869. Todas las excepciones asíncronas: entornos y nombres de constante relacionados

Entorno	Valor
JMS herramienta de administración	ALL
Programático	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
IBM MQ Explorer	Todo

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Solo las excepciones que indican una conexión interrumpida se envían al ExceptionListener. Cualquier otra excepción que se produce durante el proceso asíncrono no se notifica al ExceptionListener y, por lo tanto, la aplicación no está informada de estas excepciones. Este es el valor predeterminado a partir de IBM MQ 8.0.0 Fix Pack 2. Consulte JMS: Cambia el escucha de excepciones en IBM MQ 8.0.

Tabla 870. Excepciones que indican una conexión interrumpida: entornos y nombres de constante relacionados

Entorno	Valor
JMS herramienta de administración	CONNECTIONBROKEN
Programático	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
IBM MQ Explorer	Conexión interrumpida

Se define la siguiente constante adicional:

- Desde IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN

- Antes de IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Conceptos relacionados

Excepciones en IBM MQ classes for JMS

BROKERCCDURSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción duradera para un ConnectionConsumer.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : BROKERCCDURSUBQ

Nombre abreviado de la herramienta de administración de JMS : CCDSUB

Acceso programático

Métodos setter/getters

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Valores

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida

BROKERCCSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción no duradera para un ConnectionConsumer.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERCCSUBQ

Nombre abreviado de la herramienta de administración de JMS : CCSUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Valores

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida

BROKERCONQ

El nombre de cola de control del intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERCONQ

Nombre abreviado de la herramienta de administración de JMS : BCON

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Valores

SYSTEM.BROKER.CONTROL.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida

BROKERDURSUBQ

Cuando los IBM MQ classes for JMS se están utilizando en la modalidad de migración del proveedor de mensajería de IBM MQ , esta propiedad especifica el nombre de la cola de la que se recuperan los mensajes de suscripción duradera.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : BROKERDURSUBQ

Nombre abreviado de la herramienta de administración de JMS : BDSUB

Acceso programático

Métodos setter/getters

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Valores

SYSTEM.JMS.D.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida
Empezando por SYSTEM.JMS.D

Tareas relacionadas
Configuración de la propiedad JMS **PROVIDERVERSION**

BROKERPUBQ

El nombre de la cola donde se envían los mensajes publicados (el valor de la corriente de datos).

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERPUBQ

Nombre abreviado de la herramienta de administración de JMS : BPUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Valores

SYSTEM.BROKER.DEFAULT.STREAM
Éste es el valor predeterminado.

Cualquier serie válida

BROKERPUBQMGR

Nombre del gestor de colas al que pertenece la cola a la que se envían los mensajes publicados en el tema.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : BROKERPUBQMGR

Nombre abreviado de la herramienta de administración de JMS : BPQM

Acceso programático

Métodos setter/getters

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Valores

null
Éste es el valor predeterminado.

Cualquier serie válida

BROKERQMGR

Nombre del gestor de colas en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERQMGR

Nombre abreviado de la herramienta de administración de JMS : BQM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Valores

null
Éste es el valor predeterminado.

Cualquier serie válida

BROKERSUBQ

Cuando los IBM MQ classes for JMS se están utilizando en la modalidad de migración del proveedor de mensajería de IBM MQ , esta propiedad especifica el nombre de la cola de la que se recuperan los mensajes de suscripción no duradera.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERSUBQ

Nombre abreviado de la herramienta de administración de JMS : BSUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Valores

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida
Empezando por SYSTEM.JMS.ND

Tareas relacionadas

Configuración de la propiedad JMS **PROVIDERVERSION**

BROKERVER

La versión del intermediario que se está utilizando.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERVER

Nombre abreviado de la herramienta de administración de JMS : BVER

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerVersión ()
- MQConnectionFactory.getBrokerVersión ()

Valores

V1
Para utilizar un intermediario de publicación/suscripción de IBM MQ , o para utilizar un intermediario de IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o

WebSphere Business Integration Message Broker en modalidad de compatibilidad. Este es el valor predeterminado si TRANSPORT se establece en BIND o CLIENT.

V2
Para utilizar un intermediario de IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker en modalidad nativa. Este es el valor predeterminado si TRANSPORT se establece en DIRECT o DIRECTHTTP.

no especificada
Después de que el intermediario haya migrado de V6 a V7, establezca esta propiedad para que las cabeceras RFH2 ya no se utilicen. Después de la migración, esta propiedad ya no es relevante.

CCDTURL

Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de definición de canal de cliente y especifica cómo se puede acceder al archivo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CCDTURL

Nombre abreviado de la herramienta de administración de JMS : CCDT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Valores

null
Éste es el valor predeterminado.

Un localizador universal de recursos (URL)

CCSID

Para las fábricas de conexiones, esta propiedad especifica el ID de juego de caracteres codificado (CCSID) que se utilizará para los flujos de datos internos con el gestor de colas. Para los destinos, la propiedad define el CCSID que se utilizará para codificar datos de serie en MapMessages, StreamMessagesy TextMessages colocados en ese destino.

Nota: Normalmente no es necesario cambiar esta propiedad para las fábricas de conexiones.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CCSID

Nombre abreviado de la herramienta de administración de JMS : CCS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCCSID()

- MQConnectionFactory.getCCSID()

Valores

819
El valor predeterminado para una fábrica de conexiones.

1208
Valor predeterminado para un destino.

Cualquier entero positivo

Conceptos relacionados
Conversión de mensajes JMS

CHANNEL

El nombre del canal de conexión de cliente que se está utilizando.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CHANNEL

Nombre abreviado de la herramienta de administración de JMS : CHAN

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Valores

SYSTEM.DEF.SVRCONN
Éste es el valor predeterminado.

Cualquier serie válida

CLEANUP

Nivel de limpieza para almacenes de suscripciones BROKER o MIGRATE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLEANUP

Nombre abreviado de la herramienta de administración de JMS : CL

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCleanupNivel ()
- MQConnectionFactory.getCleanupNivel ()

Valores

SAFE
Utilice una limpieza segura. Éste es el valor predeterminado.

ASPROP
Utilice una limpieza segura, fuerte o nula según una propiedad establecida en la línea de mandatos de Java .

NINGUNO
No utilizar limpieza.

STRONG
Utilice una limpieza fuerte.

CLEANUPINT

El intervalo, en milisegundos, entre ejecuciones en segundo plano del programa de utilidad de limpieza de publicación/suscripción.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLEANUPINT

Nombre abreviado de la herramienta de administración de JMS : CLINT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCleanupIntervalo ()
- MQConnectionFactory.getCleanupIntervalo ()

Valores

3600000
Éste es el valor predeterminado.

Cualquier entero positivo

ConnectionNameList

Lista de nombres de conexión TCP/IP. La lista se intenta en orden, una vez por cada reintento de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CONNECTIONNAMELIST

Nombre abreviado de la herramienta de administración de JMS : CNLIST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setconnectionNameList ()
- MQConnectionFactory.getconnectionNameLista ()

Valores

Lista separada por comas de HOSTNAME (PORT). HOSTNAME puede ser un nombre DNS o una dirección IP.

PORT toma el valor predeterminado de 1414.

CLIENTRECONNECTOPTIONS

Opciones que rigen la reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTRECONNECTOPTIONS

Nombre abreviado de la herramienta de administración de JMS : CROPT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Valores

QMGR

La aplicación puede volver a conectarse al mismo gestor de colas al que está conectada originalmente.

Se devuelve un error con el código de razón `MQRC_RECONNECT_QMID_MISMATCH` si el gestor de colas al que la aplicación intenta conectarse, tal como se especifica en la lista de nombres de conexión, tiene un QMID diferente al gestor de colas al que se conectó originalmente.

Utilice este valor si una aplicación se puede reconectar, pero existe una afinidad entre la aplicación IBM MQ classes for JMS y el gestor de colas con el que estableció por primera vez una conexión.

Elija este valor si desea que una aplicación se vuelva a conectar automáticamente a la instancia en espera de un gestor de colas de alta disponibilidad.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

CUALQUIERA

La aplicación puede volver a conectarse a cualquiera de los gestores de colas especificados en la lista de nombres de conexión.

Utilice la opción de reconexión sólo si no hay afinidad entre las clases IBM MQ para la aplicación JMS y el gestor de colas con el que estableció inicialmente una conexión.

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT.

DISABLED

La aplicación no se reconectará.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

Si la aplicación se volverá a conectar automáticamente depende del valor del atributo de canal de IBM MQ DefReconnect.

Éste es el valor predeterminado.

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Tiempo antes de que cesen los reintentos de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTRECONNECTTIMEOUT

Nombre abreviado de la herramienta de administración de JMS : CRT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Valores

Intervalo en segundos. Valor predeterminado 1800 (30 minutos).

CLIENTID

El identificador de cliente se utiliza para identificar de forma exclusiva la conexión de aplicación para las suscripciones duraderas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTID

Nombre abreviado de la herramienta de administración de JMS : CID

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientId ()
- MQConnectionFactory.getClientId ()

Valores

null
Éste es el valor predeterminado.

Cualquier serie válida

CLONESUPP

Determina si dos o más instancias del mismo suscriptor de tema duradero se pueden ejecutar de forma simultánea.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLONESUPP

Nombre abreviado de la herramienta de administración de JMS : CLS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCloneSoporte ()
- MQConnectionFactory.getCloneSoporte ()

Valores

DISABLED
Sólo se puede ejecutar una instancia de un suscriptor de tema duradero a la vez. Éste es el valor predeterminado.

ENABLED
Dos o más instancias del mismo suscriptor de tema duradero pueden ejecutarse simultáneamente, pero cada instancia debe ejecutarse en una máquina virtual Java (JVM) independiente.

COMPHDR

Lista de las técnicas que se pueden utilizar para comprimir datos de cabecera en una conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : COMPHDR

Nombre abreviado de la herramienta de administración de JMS : HC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Valores

NINGUNO
Éste es el valor predeterminado.

SYSTEM
Se realiza la compresión de cabecera de mensaje RLE.

COMPMSG

Lista de las técnicas que se pueden utilizar para comprimir datos de mensaje en una conexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : COMPMSG

Nombre abreviado de la herramienta de administración de JMS : MC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Valores

NINGUNO
Éste es el valor predeterminado.

Una lista de uno o varios de los valores siguientes separados por caracteres en blanco:
RLE ZLIBFAST ZLIBHIGH

CONNOPT

Controla cómo se conectan al gestor de colas las aplicaciones IBM MQ classes for JMS que utilizan el transporte de enlaces.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

Nombre largo de la herramienta de administración de JMS : CONNOPT

Nombre abreviado de la herramienta de administración de JMS : CNOPT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMQConnectionOpciones ()
- MQConnectionFactory.getMQConnectionOpciones ()

Valores

STANDARD
La naturaleza del enlace entre la aplicación y el gestor de colas depende del valor del atributo *DefaultBindTipo* del gestor de colas. El valor STANDARD se correlaciona con IBM MQ *ConnectOption* MQCNO_STANDARD_BINDING.

SHARED
La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes pero comparten algunos recursos. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_SHARED_BINDING.

Aislado
La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes y no comparten recursos. El valor AISLADO se correlaciona con IBM MQ *ConnectOption* MQCNO_ISOLATED_BINDING.

FASTPATH
La aplicación y el agente del gestor de colas local se ejecutan en la misma unidad de ejecución. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_FASTPATH_BINDING.

SERIALQM
La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG
La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. El valor SERIALQSG se correlaciona con IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG.

RESTRICTQM
La aplicación solicita el uso compartido del código de conexión, pero hay restricciones en el uso compartido del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

RESTRICTQSG
La aplicación solicita el uso compartido del código de conexión, pero existen restricciones sobre el uso compartido del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG.

Para obtener más información sobre las opciones de conexión de IBM MQ , consulte Conexión a un gestor de colas utilizando la llamada MQCONNX.

CONNTAG

Código que el gestor de colas asocia con los recursos actualizados por la aplicación dentro de una unidad de trabajo mientras la aplicación está conectada al gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CONNTAG

Nombre abreviado de la herramienta de administración de JMS : CNTAG

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setConnEtiqueta ()
- MQConnectionFactory.getConnEtiqueta ()

Valores

Una matriz de bytes de 128 elementos, donde cada elemento es 0
Éste es el valor predeterminado.

Cualquier serie
El valor se trunca si tiene más de 128 bytes.

description

Una descripción del objeto almacenado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : DESCRIPTION

Nombre abreviado de la herramienta de administración de JMS : DESC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Valores

null
: Éste es el valor predeterminado.

Cualquier serie válida
: `

DIRECTAUTH

Indica si se utiliza la autenticación TLS en una conexión en tiempo real con un intermediario.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : DIRECTAUTH

Nombre abreviado de la herramienta de administración de JMS : DAUTH

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory.getDirectAuth ()

Valores

BASIC
: Sin autenticación, autenticación de nombre de usuario o autenticación de contraseña. Éste es el valor predeterminado.

CERTIFICATE
: Autenticación de certificado de clave pública.

ENCODING

Cómo se representan los datos numéricos en el cuerpo de un mensaje cuando se envía el mensaje a este destino. La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : ENCODING

Nombre abreviado de la herramienta de administración de JMS : ENC

Acceso programático

Métodos setter/getters

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Valores

La propiedad de codificación

Los valores válidos que puede tomar la propiedad ENCODING se construyen a partir de las tres subpropiedades:

Codificación de enteros
Normal o invertido

Codificación decimal
Normal o invertido

codificación de coma flotante
IEEE normal, IEEE invertido o z/OS

La propiedad ENCODING se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS
- El primer carácter representa *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad ENCODING .

Hay un valor adicional, la serie NATIVE, que establece los valores de codificación adecuados para la plataforma Java .

Los ejemplos siguientes muestran combinaciones válidas para ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

EXPIRY

La hora después de la cual caducan los mensajes en un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : EXPIRE

Nombre abreviado de la herramienta de administración de JMS : EXP

Acceso programático

Métodos setter/getters

- MQDestination.setExpiry()
- MQDestination.getExpiry()

Valores

APP
La aplicación JMS puede definir la caducidad. Éste es el valor predeterminado.

UNLIM
No se produce ninguna caducidad.

0
No se produce ninguna caducidad.

Cualquier entero positivo que represente la caducidad en milisegundos.

FAILIFQUIESCE

Esta propiedad determina si las llamadas a determinados métodos fallan si el gestor de colas está en un estado de inmovilización, o si una aplicación se está conectando a un gestor de colas utilizando el transporte CLIENT y el canal que la aplicación está utilizando se ha puesto en un estado de inmovilización, por ejemplo, utilizando el mandato **STOP CHANNEL** o **STOP CHANNEL MODE(QUIESCE)** MQSC.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : FAILIFQUIESCE

Nombre abreviado de la herramienta de administración de JMS : FIQ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Valores

SÍ
Las llamadas a determinados métodos fallan si el gestor de colas está en un estado de desactivación temporal o si el canal que se utiliza para conectarse a un gestor de colas está desactivado temporalmente. Si una aplicación detecta cualquiera de estas condiciones, la aplicación puede completar su tarea inmediata y cerrar la conexión, permitiendo que se detenga el gestor de colas o la instancia de canal. Éste es el valor predeterminado.

No
Ninguna llamada de método falla porque el gestor de colas, o el canal que se está utilizando para conectarse a un gestor de colas, está en un estado de desactivación temporal. Si especifica este valor, una aplicación no puede detectar que el gestor de colas o el canal se está desactivando temporalmente. La aplicación puede continuar realizando operaciones en el gestor de colas y, por lo tanto, impedir que se detenga el gestor de colas.

HOSTNAME

Para una conexión con un gestor de colas, el nombre de host o la dirección IP del sistema en el que se ejecuta el gestor de colas o, para una conexión en tiempo real con un intermediario, el nombre de host o la dirección IP del sistema en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : HOSTNAME

Nombre abreviado de la herramienta de administración de JMS : HOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setHostNombre ()
- MQConnectionFactory.getHostNombre ()

Valores

localhost
Éste es el valor predeterminado.

Cualquier serie válida

LOCALADDRESS

Para una conexión con un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : LOCALADDRESS

Nombre abreviado de la herramienta de administración de JMS : LA

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setLocalDirección ()
- MQConnectionFactory.getLocalDirección ()

Valores

"" (serie vacía)
Éste es el valor predeterminado.

Una serie con el formato [ip-addr] [(low-port [, high port])]
A continuación, se detallan algunos ejemplos:

```
192.0.2.0
```

El canal se enlaza a la dirección 192.0.2.0 localmente.

```
192.0.2.0(1000)
```

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza el puerto 1000.

`192.0.2.0(1000,2000)`

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza un puerto en el rango de 1000 a 2000.

`(1000)`

El canal se enlaza al puerto 1000 localmente.

`(1000,2000)`

El canal se enlaza a un puerto en el rango de 1000 a 2000 localmente.

Puede especificar un nombre de host en lugar de una dirección IP. Para una conexión en tiempo real con un intermediario, esta propiedad sólo es relevante cuando se utiliza la multidifusión, y el valor de la propiedad no debe contener un número de puerto o un rango de números de puerto. Los únicos valores válidos de la propiedad en este caso son null, una dirección IP o un nombre de host.

ESTILO de nombre de mapa

Permite utilizar el estilo de compatibilidad para los nombres de elemento MapMessage .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MAPNAMESTYLE

Nombre abreviado de la herramienta de administración de JMS : MNST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Valores

STANDARD
Se utilizará el formato de denominación de elementos estándar com.ibm.jms.JMSMapMessage . Este es el valor predeterminado y permite que se utilicen identificadores Java no legales como nombre de elemento.

Compatible
Se utilizará el formato de denominación de elementos com.ibm.jms.JMSMapMessage más antiguo. Sólo se pueden utilizar identificadores Java legales como nombre de elemento. Esto sólo es necesario si se envían mensajes de correlación a una aplicación que utiliza una versión de IBM MQ classes for JMS anterior a 5.3.

MAXBUFFSIZE

Número máximo de mensajes recibidos que se pueden almacenar en un almacenamiento intermedio de mensajes interno mientras se espera a que la aplicación los procese. Esta propiedad sólo se aplica cuando TRANSPORT tiene el valor DIRECT o DIRECTHTTP.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : MAXBUFFSIZE

Nombre abreviado de la herramienta de administración de JMS : MBSZ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Valores

1000
Éste es el valor predeterminado.

Cualquier entero positivo

MDREAD

Esta propiedad determina si una aplicación JMS puede extraer los valores de los campos MQMD.

Objetos aplicables

Nombre largo de la herramienta de administración de JMS : MDREAD

Nombre abreviado de la herramienta de administración de JMS : MDR

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Valores

No
Al enviar mensajes, las propiedades JMS_IBM_MQMD* en un mensaje enviado no se actualizan para reflejar los valores de campo actualizados en el MQMD. Al recibir mensajes, ninguna de las propiedades JMS_IBM_MQMD* está disponible para un mensaje recibido, incluso si el emisor ha establecido todas o algunas de ellas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

Sí
Al enviar mensajes, todas las propiedades JMS_IBM_MQMD* de un mensaje enviado se actualizan para reflejar los valores de campo actualizados en MQMD, incluidas las propiedades que el remitente no ha establecido explícitamente. Al recibir mensajes, todas las propiedades JMS_IBM_MQMD* están disponibles en un mensaje recibido, incluidas las propiedades que el remitente no ha establecido explícitamente.

Para los programas, utilice True.

MDWRITE

Esta propiedad determina si una aplicación JMS puede establecer los valores de los campos MQMD.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : MDWRITE

Nombre abreviado de la herramienta de administración de JMS : MDR

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Valores

No

Todas las propiedades JMS_IBM_MQMD* se ignoran y sus valores no se copian en la estructura MQMD subyacente. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

SÍ

Se procesan las propiedades JMS_IBM_MQMD*. Sus valores se copian en la estructura MQMD subyacente.

Para los programas, utilice True.

MDMSGCTX

Determina el nivel de contexto de mensaje que debe ser establecido por la aplicación de JMS. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.

Objetos aplicables

Nombre largo de la herramienta de administración de JMS : MDMSGCTX

Nombre abreviado de la herramienta de administración de JMS : MDCTX

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Valores

DEFAULT

La llamada de API MQOPEN y la estructura MQPMO no especifican opciones de contexto de mensaje explícitas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT

La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_IDENTITY_CONTEXT y la estructura MQPMO especificaMQPMO_SET_IDENTITY_CONTEXT.

Para los programas, utilice WMQ_MDCTX_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT

La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_ALL_CONTEXT y la estructura MQPMO especifica MQPMO_SET_ALL_CONTEXT.

Para programas, utilice WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Número máximo de mensajes que se deben tomar de una cola en un paquete cuando se utiliza la entrega de mensajes asíncrona.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MAXBUFFSIZE

Nombre abreviado de la herramienta de administración de JMS : MBSZ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Valores

10
Éste es el valor predeterminado.

Cualquier entero positivo

MSGBODY

Determina si una aplicación JMS accede a la MQRFH2 de un mensaje IBM MQ como parte de la carga útil del mensaje.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : WMQ_MESSAGE_BODY

Nombre abreviado de la herramienta de administración de JMS : MBODY

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Valores

SIN ESPECIFICAR
En el envío, determina si IBM MQ classes for JMS genera e incluye una cabecera MQRFH2, dependiendo del valor de WMQ_TARGET_CLIENT. Al recibir, actúa como valor JMS.

JMS
En el envío, IBM MQ classes for JMS genera automaticamente una cabecera MQRFH2 y la incluye en el mensaje de IBM MQ.

En la recepción, IBM MQ classes for JMS establece las propiedades de mensajes de JMS de acuerdo con los valores contenidos en la cabecera MQRFH2 (si existe); no presenta la cabecera MQRFH2 como parte del cuerpo del mensaje JMS.

MQ
En el envío, IBM MQ classes for JMS no genera una cabecera MQRFH2.

En la recepción, IBM MQ classes for JMS presenta MQRFH2 como parte del cuerpo del mensaje de JMS.

MSGRETENTION

Indica si el consumidor de conexión mantiene los mensajes no entregados en la cola de entrada.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Nombre largo de la herramienta de administración de JMS : MSGRETENTION

Nombre abreviado de la herramienta de administración de JMS : MRET

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMessageRetención ()
- MQConnectionFactory.getMessageRetención ()

Valores

Sí
Los mensajes no entregados permanecen en la cola de entrada. Éste es el valor predeterminado.

No
Los mensajes no entregados se tratan de acuerdo con sus opciones de disposición.

MSGSELECTION

Determina si la selección de mensajes la realiza el IBM MQ classes for JMS o el intermediario. Si TRANSPORT tiene el valor DIRECT, el intermediario siempre realiza la selección de mensajes y se ignora el valor de MSGSELECTION. La selección de mensajes por parte del intermediario no está soportada cuando BROKERVER tiene el valor V1.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MSGSELECTION

Nombre abreviado de la herramienta de administración de JMS : MSEL

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMessageSelection ()
- MQConnectionFactory.getMessageSelección ()

Valores

CLIENT
La selección de mensajes la realiza IBM MQ classes for JMS. Éste es el valor predeterminado.

BROKER
La selección de mensajes ha sido realizada por el intermediario.

MULTICAST

Para habilitar la multidifusión en una conexión en tiempo real con un intermediario y, si está habilitada, para especificar la forma precisa en la que se utiliza la multidifusión para entregar mensajes del intermediario a un consumidor de mensajes. La propiedad no tiene ningún efecto sobre cómo un productor de mensajes envía mensajes a un intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Tema

Nombre largo de la herramienta de administración de JMS : MULTICAST

Nombre abreviado de la herramienta de administración de JMS : MCAST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Valores

DISABLED
Los mensajes no se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Este es el valor predeterminado para los objetos ConnectionFactory y TopicConnectionFactory.

ASCF
Los mensajes se entregan a un consumidor de mensajes de acuerdo con el valor de multidifusión para la fábrica de conexiones asociada al consumidor de mensajes. El valor de multidifusión para la fábrica de conexiones se anota en el momento en que se crea el consumidor de mensajes. Este valor sólo es válido para los objetos de tema y es el valor predeterminado para los objetos de tema.

ENABLED
Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Se utiliza una calidad de servicio fiable si el tema está configurado para la multidifusión fiable.

PROVEEDOR
Si el tema está configurado para multidifusión fiable en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión con una calidad de servicio fiable. Si el tema no está configurado para la multidifusión fiable, no puede crear un consumidor de mensajes para el tema.

NOTR
Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión. No se utiliza una calidad de servicio fiable, aunque el tema se haya configurado para la multidifusión fiable.

OPTIMISTICPUBLICATION

Esta propiedad determina si IBM MQ classes for JMS devuelve el control inmediatamente a un publicador que ha publicado un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al publicador.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : OPTIMITICPUBLICATION

Nombre abreviado de la herramienta de administración de JMS : OPTPUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setOptimisticPublicación ()
- MQConnectionFactory.getOptimisticPublicación ()

Valores

No
Cuando un publicador publica un mensaje, IBM MQ classes for JMS no devuelve el control al publicador hasta que haya completado todo el proceso asociado a la llamada y pueda informar del resultado al publicador. Éste es el valor predeterminado.

SÍ
Cuando un publicador publica un mensaje, IBM MQ classes for JMS devuelve el control al publicador inmediatamente, antes de que haya completado todo el proceso asociado a la llamada y pueda informar del resultado al publicador. IBM MQ classes for JMS informa del resultado sólo cuando el publicador confirma el mensaje.

OUTCOMENOTIFICATION

Esta propiedad determina si IBM MQ classes for JMS devuelve el control inmediatamente a un suscriptor que acaba de reconocer o confirmar un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : OUTCOMENOTIFICACIÓN

Nombre abreviado de la herramienta de administración de JMS : NOTIFY

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setOutcomeNotificación ()
- MQConnectionFactory.getOutcomeNotificación ()

Valores

SÍ
Cuando un suscriptor reconoce o confirma un mensaje, IBM MQ classes for JMS no devuelve el control al suscriptor hasta que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor. Éste es el valor predeterminado.

No
Cuando un suscriptor reconoce o confirma un mensaje, IBM MQ classes for JMS devuelve el control al suscriptor inmediatamente, antes de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor.

PERSISTENCE

Persistencia de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : PERSISTENCE

Nombre abreviado de la herramienta de administración de JMS : PER

Acceso programático

Métodos setter/getters

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Valores

APP
La persistencia la define la aplicación JMS . Éste es el valor predeterminado.

qdef
La persistencia toma el valor predeterminado de la cola.

pers
Los mensajes son persistentes.

no
Los mensajes no son persistentes.

HIGH
Consulte Mensajes persistentes deJMS para obtener más información sobre el uso de este valor.

POLLINGINT

Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola. Si con frecuencia sucede esto de que no haya disponible ningún mensaje adecuado para ninguno de los escuchas de mensajes de una sesión, considere aumentar el valor de esta propiedad. Esta propiedad sólo tiene relevancia si TRANSPORT tiene el valor BIND o CLIENT.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : POLLINGINT

Nombre abreviado de la herramienta de administración de JMS : PINT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setPollingIntervalo ()
- MQConnectionFactory.getPollingIntervalo ()

Valores

5000
Éste es el valor predeterminado.

Cualquier entero positivo

PORT

Para una conexión con un gestor de colas, el número del puerto en el que el gestor de colas está a la escucha o, para una conexión en tiempo real con un intermediario, el número del puerto en el que el intermediario está a la escucha de conexiones en tiempo real.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PORT

Nombre abreviado de la herramienta de administración de JMS : PORT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

Valores

1414
Este es el valor predeterminado si TRANSPORT se establece en CLIENT.

1506
Este es el valor predeterminado si TRANSPORT se establece en DIRECT o DIRECTHTTP.

Cualquier entero positivo

PRIORITY

Prioridad de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : PRIORITY

Nombre abreviado de la herramienta de administración de JMS : PRI

Acceso programático

Métodos setter/getters

- MQDestination.setPriority()
- MQDestination.getPriority()

Valores

APP
La prioridad la define la aplicación JMS . Éste es el valor predeterminado.

qdef
La prioridad toma el valor predeterminado de la cola.

Cualquier entero en el rango 0-9
De menor a mayor.

PROCESSDURATION

Esta propiedad determina si un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a IBM MQ classes for JMS.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : PROCESSDURATION

Nombre abreviado de la herramienta de administración de JMS : PROCDUR

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setProcessDuración ()
- MQConnectionFactory.getProcessDuración ()

Valores

DESCONOCIDO
Un suscriptor no puede dar ninguna garantía sobre la rapidez con la que puede procesar cualquier mensaje que reciba. Éste es el valor predeterminado.

SHORT
Un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a IBM MQ classes for JMS.

PROVIDERVERSION

Esta propiedad diferencia entre las tres modalidades de operación de mensajería de IBM MQ : modalidad normal de proveedor de mensajería de IBM MQ , modalidad normal de proveedor de mensajería de IBM MQ con restricciones y modalidad de migración de proveedor de mensajería de IBM MQ .

La modalidad normal del proveedor de mensajería de IBM MQ utiliza todas las características de un gestor de colas de IBM MQ para implementar JMS. Esta modalidad está optimizada para utilizar la API y la funcionalidad de JMS 2.0. La modalidad normal con restricciones del proveedor de mensajería de IBM MQ utiliza la API JMS 2.0 , pero no las nuevas características como, por ejemplo, suscripciones compartidas, entrega retardada o envío asíncrono.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROVIDERVERSION

Nombre abreviado de la herramienta de administración de JMS : PVER

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setProviderVersión ()
- MQConnectionFactory.getProviderVersión ()

Valores

Puede establecer la propiedad **PROVIDERVERSION** en cualquiera de los valores 8 (modalidad normal), 7 (modalidad normal con restricciones), 6 (modalidad de migración) o `unspecified` (el valor predeterminado). El valor que especifique para la propiedad **PROVIDERVERSION** debe ser una serie. Si va a especificar la opción 8, 7 o 6, puede hacerlo en cualquiera de los siguientes formatos:

- V.R.M.F
- V.R.M
- V.R
- V

donde V, R, M y F son valores enteros mayores que o iguales a cero. Los valores R, M y F adicionales son opcionales y están disponibles para su uso en caso de que se necesite un control de grano fino. Por ejemplo, si desea utilizar un nivel **PROVIDERVERSION** de 7, puede establecer **PROVIDERVERSION**=7, `7.0`, `7.0.0` o `7.0.0.0`.

8 - Modalidad normal

La aplicación JMS utiliza la modalidad normal del proveedor de mensajería IBM MQ. La modalidad normal utiliza todas las características de un gestor de colas IBM MQ para implementar JMS. Esta modalidad se optimiza para utiliza la API y las funciones de JMS 2.0.

Si se conecta a un gestor de colas con un nivel de mandato de 800, se pueden utilizar todas las API y características de JMS 2.0, por ejemplo envío asíncrono, entrega retrasada o suscripción compartida.

Si el gestor de colas especificado en los valores de la fábrica de conexiones no es un gestor de colas IBM MQ 8.0.0, `createConnection` falla con una excepción `JMSFMQ0003`.

La modalidad normal del proveedor de mensajería de IBM MQ utiliza la característica de conversaciones compartidas y el número de conversaciones que se pueden compartir lo controla la propiedad **SHARECNV()** en el canal de conexión del servidor. Si esta propiedad se establece en `0`, no puede utilizar la modalidad normal del proveedor de mensajería de IBM MQ y el método `createConnection` falla con una excepción `JMSCC5007`.

7 - Modalidad normal con restricciones

La aplicación JMS utiliza la modalidad normal con restricciones del proveedor de mensajería IBM MQ. Esta modalidad utiliza la API de JMS 2.0, pero no las nuevas características tales como suscripciones compartidas, entrega retrasada o envío asíncrono.

Si establece **PROVIDERVERSION** en 7 solo está disponible el proveedor de mensajería IBM MQ normal con la modalidad de restricciones de operación. Si el gestor de colas especificado en los valores de la fábrica de conexiones no es un gestor de colas IBM WebSphere MQ 7.0.1 o posterior, el método `createConnection` falla con la excepción `JMSFCC5008`.

Si se conecta utilizando la modalidad normal con restricciones a un gestor de colas con un nivel de mandatos entre 700 y 800, puede utilizar la API de JMS 2.0, pero no las características de envío asíncrono, entrega retrasada o suscripción compartida.

La modalidad normal del proveedor de mensajería de IBM MQ con restricciones utiliza la característica de compartición de conversaciones y el número de conversaciones que se pueden compartir lo controla la propiedad **SHARECNV()** en el canal de conexión del servidor. Si esta propiedad se establece a `0`, no se podrá usar el modo normal de mensajería con restricciones de IBM MQ, y el método `createConnection` fallará con la excepción `JMSCC5007`.

6 - Modalidad de migración

La aplicación JMS utiliza la modalidad de migración de proveedor de mensajería de IBM MQ.

Las IBM MQ classes for JMS utilizan las características y los algoritmos proporcionados con IBM WebSphere MQ 6.0. Si desea conectar a WebSphere Message Broker 6.0 o 6.1 utilizando IBM WebSphere MQ Enterprise Transport 6.0, debe utilizar esta modalidad. Puede conectarse a un gestor de colas de IBM MQ 8.0 utilizando esta modalidad, pero no se utiliza ninguna de las características de un gestor de colas de IBM MQ classes for JMS, por ejemplo, la lectura anticipada o la modalidad continua.

Si tiene un cliente de IBM MQ 8.0 o posterior que se conecta a un gestor de colas de IBM MQ 8.0 o posterior, la selección de mensajes la realiza el gestor de colas en lugar del sistema cliente.

Si se especifica la modalidad de migración de proveedor de mensajería de IBM MQ e intenta utilizar cualquiera de las API JMS 2.0, la llamada de método de API falla con la excepción `JMSCC5007`.

`unspecified` (valor predeterminado)

La propiedad **PROVIDERVERSION** se establece en *unspecified* de forma predeterminada.

Una fábrica de conexiones que se ha creado con una versión anterior de IBM MQ classes for JMS en JNDI toma este valor cuando se utiliza la fábrica de conexiones con la nueva versión de IBM MQ classes for JMS. El siguiente algoritmo se utiliza para determinar qué modalidad de

operación se utiliza. Este algoritmo se utiliza cuando se llama al método `createConnection` y utiliza otros aspectos de la fábrica de conexiones para determinar si se necesita la modalidad normal de proveedor de mensajería de IBM MQ, la modalidad normal con restricciones o la modalidad de migración de proveedor de mensajería de IBM MQ.

1. En primer lugar, se realiza un intento de utilizar la modalidad normal de proveedor de mensajería de IBM MQ.
2. Si el gestor de colas conectado no es IBM MQ 8.0 o posterior, se realiza un intento de utilizar la modalidad normal con restricciones del proveedor de mensajería de IBM MQ.
3. Si el gestor de colas conectado no es IBM WebSphere MQ 7.0.1 o posterior, la conexión se cierra y se utiliza en su lugar la modalidad de migración de proveedor de mensajería de IBM MQ.
4. Si la propiedad **SHARECNV** del canal de conexión del servidor se establece en 0, la conexión se cierra y en su lugar se utiliza la modalidad de migración del proveedor de mensajería de IBM MQ.
5. Si **BROKERVER** se establece en V1 o el valor predeterminado de *unspecified*, se sigue utilizando la modalidad normal del proveedor de mensajería de IBM MQ y, por lo tanto, las operaciones de publicación/suscripción utilizan las características nuevas IBM WebSphere MQ 7.0.1 o posteriores.

   Consulte ALTER QMGR para obtener información sobre el parámetro PSMODE del mandato ALTER QMGR si necesita más información sobre compatibilidad.
6. Si **BROKERVER** se establece en V2, la acción realizada depende del valor de **BROKERQMGR**:
   - Si el **BROKERQMGR** `está en blanco`:

     Si la cola especificada mediante la propiedad **BROKERCONQ** se puede abrir para la salida (es decir, `MQOPEN` para la salida resulta satisfactorio) y **PSMODE** en el gestor de colas se establece en `COMPAT` o `DISABLED`, se utiliza la modalidad de migración del proveedor de mensajería de IBM MQ.
   - Si la cola especificada por la propiedad **BROKERCONQ** no se puede abrir para la salida, o el atributo **PSMODE** se establece en `ENABLED`:

     Se utiliza la modalidad normal de proveedor de mensajería de IBM MQ.
   - Si **BROKERQMGR** `no está en blanco`:

     Se utiliza la modalidad de proveedor de mensajería de IBM MQ.

Si no puede cambiar la fábrica de conexiones que está utilizando, puede utilizar la propiedad `com.ibm.msg.client.wmq.overrideProviderVersion` para alterar temporalmente cualquier valor de la fábrica de conexiones. Esta alteración temporal se aplica a todas las fábricas de conexiones de la JVM, pero los objetos de fábrica de conexiones reales no se modifican.

Tareas relacionadas

Configuración de la propiedad JMS **PROVIDERVERSION**

PROXYHOSTNAME

Nombre de host o dirección IP del sistema en el que se ejecuta el servidor proxy cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : PROXYHOSTNAME

Nombre abreviado de la herramienta de administración de JMS : PHOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setProxyHostName()

- MQConnectionFactory.getProxyHostName()

Valores

null
Nombre de host del servidor proxy. Éste es el valor predeterminado.

PROXYPORT

Número del puerto en el que el servidor proxy está a la escucha cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : PROXYPORT

Nombre abreviado de la herramienta de administración de JMS : PPORT

Acceso programático

Métodos setter/getters

MQConnectionFactory.setProxyPuerto ()

MQConnectionFactory.getProxyPuerto ()

Valores

443
Número de puerto del servidor proxy. Éste es el valor predeterminado.

PUBACKINT

Número de mensajes publicados por un publicador antes de que IBM MQ classes for JMS solicite un acuse de recibo al intermediario.

Cuando se reduce el valor de esta propiedad, IBM MQ classes for JMS solicita acuses de recibo con más frecuencia, por lo tanto, el rendimiento del publicador disminuye. Cuando aumente el valor, IBM MQ classes for JMS tardará más tiempo en emitir una excepción si el intermediario falla. Esta propiedad sólo tiene relevancia si TRANSPORT tiene el valor BIND o CLIENT.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROXYPORT

Nombre abreviado de la herramienta de administración de JMS : PPORT

Acceso programático

Métodos setter/getters

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Valores

25
Cualquier entero positivo puede ser el valor predeterminado.

PUTASYNCALLOWED

Esta propiedad determina si se permite a los productores de mensajes utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : PUTASYNCALLOWED

Nombre abreviado de la herramienta de administración de JMS : PAALD

Acceso programático

Métodos setter/getters

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Valores

AS_DEST
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola o tema. Éste es el valor predeterminado.

AS_Q_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola.

AS_TOPIC_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de tema.

No
Las operaciones de transferencia asíncrona no están permitidas.

SÍ
Las operaciones de transferencia asíncrona están permitidas.

QMANAGER

Nombre del gestor de colas al que se va a conectar.

Sin embargo, si la aplicación utiliza una tabla de definiciones de canal de cliente para conectarse a un gestor de colas, consulte Utilización de una tabla de definiciones de canal de cliente con IBM MQ classes for JMS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : QMANAGER

Nombre abreviado de la herramienta de administración de JMS : QMGR

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setQueueGestor ()
- MQConnectionFactory.getQueueGestor ()

Valores

"" (serie vacía)
Cualquier serie puede ser el valor predeterminado.

QUEUE

El nombre del destino de cola de JMS . Coincide con el nombre de la cola utilizada por el gestor de colas.

Objetos aplicables

Cola

Nombre largo de la herramienta de administración de JMS : QUEUE

Nombre abreviado de la herramienta de administración de JMS : QU

Valores

Cualquier serie
Cualquier nombre de cola IBM MQ válido.

Conceptos relacionados
Reglas para denominar objetos de IBM MQ >

READAHEADALLOWED

Esta propiedad determina si los consumidores de mensajes y los navegadores de colas pueden utilizar la lectura anticipada para obtener mensajes no persistentes de este destino en un almacenamiento intermedio interno antes de recibirlos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : READAHEADALLOWED

Nombre abreviado de la herramienta de administración de JMS : RAALD

Acceso programático

Métodos setter/getters

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Valores

AS_DEST
Determine si se permite la lectura anticipada consultando la definición de cola o tema. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST en programas.

AS_Q_DEF
Determine si se permite la lectura anticipada consultando la definición de cola.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF en programas.

AS_TOPIC_DEF
Determine si se permite la lectura anticipada consultando la definición de tema.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF en los programas.

No

No se permite la lectura anticipada.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED en los programas.

SÍ

La lectura anticipada está permitida.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED en los programas.

READAHEADCLOSEPOLICY

Para los mensajes que se entregan a un escucha de mensajes asíncrono, qué sucede con los mensajes del almacenamiento intermedio de lectura anticipada interno cuando se cierra el consumidor de mensajes.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : READAHEADCLOSEPOLICY

Nombre abreviado de la herramienta de administración de JMS : RACP

Acceso programático

Métodos setter/getters

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Valores

DELIVER_ALL

Todos los mensajes del almacenamiento intermedio de lectura anticipada interno se entregan al escucha de mensajes de la aplicación antes de devolverlos. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_READ_AHEAD_DELIVERALL en programas.

ACTUAL_ENTREGA

Solo se completa la invocación del escucha de mensajes actual antes de la devolución, posiblemente dejando mensajes en el almacenamiento intermedio de lectura anticipada interno que, después, se descartan.

Utilice WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT en programas.

RECEIVECCSID

La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora a menos que RECEPECONVERSION se establezca en `WMQ_RECEIVE_CONVERSION_QMGR`

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : RECEIVECCSID

Nombre abreviado de la herramienta de administración de JMS : RCCS

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Valores

`WMQConstants.WMQ_RECEIVE_CCSID_JVM_DEFAULT`
0 - Utilizar `Charset.defaultCharset` de JVM

1208
`UTF-8`

ccsid
Identificador de juego de caracteres codificado soportado.

RECEIVECONVERSION

Propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : RECEPECONVERSION

Nombre abreviado de la herramienta de administración de JMS : RCNV

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Valores

`WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG`
1 -Sólo realiza la conversión de datos en el cliente JMS . El valor predeterminado de hasta V7.0, y de, e incluyendo, 7.0.1.5.

`WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR`
2 -Realizar la conversión de datos en el gestor de colas antes de enviar un mensaje al cliente. El valor predeterminado (y solo) de V7.0 a V7.0.1.4 inclusive, excepto si se aplica el APAR IC72897 .

RECEIVEISOLATION

Esta propiedad determina si un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptores.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : RECEIVEISOLATION

Nombre abreviado de la herramienta de administración de JMS : RCVISOL

Valores

CONFIRMADO
Un suscriptor sólo recibe los mensajes de la cola de suscriptores que se han confirmado. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_RCVISOL_COMMITTED en los programas.

SIN CONFIRMAR
Un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptor.

Utilice WMQConstants.WMQ_RCVISOL_UNCOMMITTED en programas.

RECEXIT

Identifica una salida de recepción de canal, o una secuencia de salidas de recepción, que se ejecutarán sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de recepción. Para obtener más información, consulte Configuración de las clases IBM MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : RECEXIT

Nombre abreviado de la herramienta de administración de JMS : RCX

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReceiveSalir ()
- MQConnectionFactory.getReceiveSalir ()

Valores

- `nulo`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
  - El nombre de una clase que implementa la interfaz `WMQReceiveExit` (para una salida de recepción de canal escrita en Java).
  - Una serie con el formato *libraryName(entryPointNombre)* (para una salida de recepción de canal no escrita en Java).

RECEXITINIT

Los datos de usuario que se pasan a las salidas de recepción de canal cuando se llaman.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : RECEXITINIT

Nombre abreviado de la herramienta de administración de JMS : RCXI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Valores

null
Una cadena que comprende uno o más elementos de datos de usuario separados por comas. Éste es el valor predeterminado.

REPLYTOSTYLE

Determina cómo se construye el campo JMSReplyTo en un mensaje recibido.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : REPLYTOSTYLE

Nombre abreviado de la herramienta de administración de JMS : RTOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Valores

DEFAULT
Equivale a `MQMD`.

RFH2
Utilice el valor proporcionado en la cabecera RFH2 . Si se ha establecido un valor JMSReplyTo en la aplicación emisora, utilice ese valor.

MQMD
Utilice el valor proporcionado por MQMD. Este comportamiento es equivalente al comportamiento predeterminado de IBM WebSphere MQ 6.0.2 Fix Pack 4 y 6.0.2.5.

Si el valor JMSReplyTo establecido por la aplicación emisora no contiene un nombre de gestor de colas, el gestor de colas receptor inserta su propio nombre en el MQMD. Si establece este parámetro en `MQMD`, la cola de respuesta que utiliza se encuentra en el gestor de colas receptor. Si establece este parámetro en RFH2, la cola de respuesta que utiliza se encuentra en el gestor de colas especificado en la RFH2 del mensaje enviado, tal como lo estableció originalmente la aplicación emisora.

Si el valor JMSReplyTo establecido por la aplicación emisora contiene un nombre de gestor de colas, el valor de este parámetro no es importante porque tanto MQMD como RFH2 contienen el mismo valor.

RESCANINT

Cuando un consumidor de mensajes del dominio punto a punto utiliza un selector de mensajes para seleccionar qué mensajes desea recibir, IBM MQ classes for JMS busque en la cola IBM MQ los mensajes adecuados en la secuencia determinada por el atributo `MsgDeliverySequence` de la cola.

Después de que IBM MQ classes for JMS encuentre un mensaje adecuado y lo entregue al consumidor, IBM MQ classes for JMS reanude la búsqueda del siguiente mensaje adecuado desde su posición actual

en la cola. IBM MQ classes for JMS continúa buscando en la cola de esta forma hasta que alcanza el final de la cola, o hasta que el intervalo de tiempo en milisegundos, determinado por el valor de esta propiedad, ha caducado. En cada caso, IBM MQ classes for JMS vuelve al principio de la cola para continuar la búsqueda y comienza un nuevo intervalo de tiempo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : RESCANINT

Nombre abreviado de la herramienta de administración de JMS : RINT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setRescanIntervalo ()
- MQConnectionFactory.getRescanIntervalo ()

Valores

5000
Cualquier entero positivo puede ser el valor predeterminado.

SECEXIT

Identifica una salida de seguridad de canal.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de seguridad. Para obtener más información, consulte Configuración de las clases IBM MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SECEXIT

Nombre abreviado de la herramienta de administración de JMS : SXC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSecuritySalir ()
- MQConnectionFactory.getSecurityExit ()

Valores

- `nulo`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
  - El nombre de una clase que implementa la interfaz `WMQSecurityExit` (para una salida de seguridad de canal escrita en Java).
  - Una serie con el formato *libraryName(entryPointNombre)* (para una salida de seguridad de canal no escrita en Java).

SECEXITINIT

Los datos de usuario que se pasan a una salida de seguridad de canal cuando se invoca.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SECEXITINIT

Nombre abreviado de la herramienta de administración de JMS : SCXI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Valores

null
Cualquier serie puede ser el valor predeterminado.

SENDCHECKCOUNT

Número de llamadas de envío que se deben permitir entre las comprobaciones de errores de transferencia asíncrona, dentro de una misma sesión no transaccional de JMS

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SENDCHECKCOUNT

Nombre abreviado de la herramienta de administración de JMS : SCC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Valores

null
Cualquier serie puede ser el valor predeterminado.

SENDEXIT

Identifica una salida de envío de canal o una secuencia de salidas de envío que se van a ejecutar sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de envío. Para obtener más información, consulte Configuración de las clases IBM MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SENDEXIT

Nombre abreviado de la herramienta de administración de JMS : SDX

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSendSalir ()
- MQConnectionFactory.getSendSalir ()

Valores

- `nulo`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
  - El nombre de una clase que implementa la interfaz `WMQSendExit` (para una salida de emisión de canal escrita en Java).
  - Una serie con el formato *libraryName(entryPointNombre)* (para una salida de emisión de canal no escrita en Java).

SENDEXITINIT

Los datos de usuario que se pasan a las salidas de emisión de canal cuando se invocan.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SENDEXITINIT

Nombre abreviado de la herramienta de administración de JMS : SDXI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

Valores

null
: Cualquier serie que comprenda uno o más elementos de datos de usuario separados por comas puede ser el valor predeterminado.

SHARECONVALLOWED

Para las aplicaciones que utilizan la modalidad normal o la modalidad normal de proveedor de mensajería de IBM MQ con restricciones, esta propiedad determina si la función de compartición de conversaciones se utiliza para conexiones, sesiones y contextos de JMS creados a partir de la fábrica de conexiones.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SHARECONVALLOWED

Nombre abreviado de la herramienta de administración de JMS : SCALD

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Valores

SÍ

Las conexiones, sesiones y contextos de JMS que se crean a partir de la fábrica de conexiones dentro de la misma JVM pueden compartir una instancia de canal (que se correlaciona con una conexión TCP/IP) cuando sea apropiado.

Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES.

NO

Cada conexión JMS creada a partir de la fábrica de conexiones, y cada sesión de JMS que se crea a partir de esas conexiones de JMS, tiene su propia instancia de canal (conexión TCP/IP) con un gestor de colas.

Para contextos JMS , el primer contexto que se crea a partir de la fábrica de conexiones crea dos instancias de canal (conexiones TCP/IP). Otros contextos de JMS que se crean a partir del primero tienen su propia instancia de canal (conexión TCP/IP).

Para los programas, utilice WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO.

Conceptos relacionados

Modalidades de funcionamiento del proveedor de mensajería de IBM MQ

Compartición de una conexión TCP/IP en clases IBM MQ para JMS

SPARSESUBS

Controla la política de recuperación de mensajes de un objeto TopicSubscriber.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración de JMS : SPARSESUBS

Nombre abreviado de la herramienta de administración de JMS : SSUBS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSparseSuscripciones ()
- MQConnectionFactory.getSparseSuscripciones ()

Valores

No

Las suscripciones reciben mensajes coincidentes frecuentes. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice false.

SÍ

Las suscripciones reciben mensajes coincidentes poco frecuentes. Este valor exige que la cola de suscripciones se pueda abrir para examinarla.

Para los programas, utilice true.

SSLCIPHERSUITE

CipherSuite que se debe utilizar para una conexión TLS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SSLCIPHERSUITE

Nombre abreviado de la herramienta de administración de JMS : SCPHS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSSLCipherSuite ()
- MQConnectionFactory.getSSLCipherSuite ()

Valores

null

Éste es el valor predeterminado. Para obtener más información, consulte Propiedades TLS de objetos JMS.

SSLCRL

Servidores CRL para comprobar la revocación de certificados TLS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SSLCRL

Nombre abreviado de la herramienta de administración de JMS : SCRL

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSSLCertAlmacenes ()
- MQConnectionFactory.getSSLCertAlmacenes ()

Valores

null
Lista separada por espacios de URL de LDAP. Éste es el valor predeterminado. Para obtener más información, consulte Propiedades TLS de objetos JMS.

SSLFIPSREQUIRED

Esta propiedad determina si una conexión TLS debe utilizar una CipherSuite soportada por el proveedor IBM Java JSSE FIPS (IBMJSSEFIPS).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SSLFIPSREQUIRED

Nombre abreviado de la herramienta de administración de JMS : SFIPS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSSLFipsObligatorio ()
- MQConnectionFactory.getSSLFipsObligatorio ()

Valores

No
Una conexión TLS puede utilizar cualquier CipherSuite que no esté soportada por el proveedor IBM Java JSSE FIPS (IBMJSSEFIPS).

Éste es el valor predeterminado. En los programas, utilice false.

SÍ
Una conexión TLS debe utilizar una CipherSuite soportada por IBMJSSEFIPS.

En los programas, utilice true.

SSLPEERNAME

Para TLS, un esqueleto de *nombre distinguido* que debe coincidir con el proporcionado por el gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SSLPEERNAME

Nombre abreviado de la herramienta de administración de JMS : SPEER

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSSLPeerNombre ()
- MQConnectionFactory.getSSLPeerNombre ()

Valores

null
Éste es el valor predeterminado. Para obtener más información, consulte Propiedades TLS de objetos JMS.

SSLRESETCOUNT

Para TLS, el número total de bytes enviados y recibidos por una conexión antes de que se renegocie la clave secreta que se utiliza para el cifrado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SSLRESETCOUNT

Nombre abreviado de la herramienta de administración de JMS : SRC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSSLResetRecuento ()
- MQConnectionFactory.getSSLResetRecuento ()

Valores

0
Cero, o cualquier entero positivo menor o igual que 999, 999, 999. Éste es el valor predeterminado. Para obtener más información, consulte Propiedades TLS de objetos JMS.

STATREFRESHINT

Intervalo, en milisegundos, entre las actualizaciones de la transacción de larga ejecución que detecta cuando un suscriptor pierde su conexión con el gestor de colas.

Esta propiedad sólo es relevante si SUBSTORE tiene el valor QUEUE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : STATREFRESHINT

Nombre abreviado de la herramienta de administración de JMS : SRI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Valores

60000
Cualquier entero positivo puede ser el valor predeterminado. Para obtener más información, consulte Propiedades TLS de objetos JMS.

SUBSTORE

Donde IBM MQ classes for JMS almacena datos persistentes relacionados con suscripciones activas.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SUBSTORE

Nombre abreviado de la herramienta de administración de JMS : SS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSubscriptionAlmacén ()
- MQConnectionFactory.getSubscriptionStore ()

Valores

BROKER
Utilice el almacén de suscripciones basado en intermediario para mantener los detalles de las suscripciones. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRATE
Transfiera la información de suscripción del almacén de suscripciones basado en cola al almacén de suscripciones basado en intermediario.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_MIGRATE.

QUEUE
Utilice el almacén de suscripciones basado en cola para contener los detalles de las suscripciones.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPOINTALLGETS

Esta propiedad determina si todas las obtenciones se deben realizar bajo punto de sincronismo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SYNCPOINTALLGETS

Nombre abreviado de la herramienta de administración de JMS : SPAG

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Valores

No
Éste es el valor predeterminado.

Sí

TARGCLIENT

Esta propiedad determina si el formato IBM MQ RFH2 se utiliza para intercambiar información con aplicaciones de destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : TARGCLIENT

Nombre abreviado de la herramienta de administración de JMS : TC

Acceso programático

Métodos setter/getters

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Valores

JMS

El destino del mensaje es una aplicación JMS . Este es el valor predeterminado para las herramientas administrativas.

Para programas, utilice WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ

El destino del mensaje es una aplicación que no es deJMS IBM MQ .

Para programas, utilice WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Esta propiedad determina si un mensaje de respuesta, enviado a la cola identificada por el campo de cabecera JMSReplyTo de un mensaje de entrada, tiene una cabecera MQRFH2 sólo si el mensaje de entrada tiene una cabecera MQRFH2 .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TARGCLIENTMATCHING

Nombre abreviado de la herramienta de administración de JMS : TCM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Valores

SÍ

Si un mensaje de entrada no tiene una cabecera MQRFH2 , la propiedad TARGCLIENT del objeto Queue derivado del campo de cabecera JMSReplyTo del mensaje se envía a MQ. Si el mensaje tiene una cabecera MQRFH2 , la propiedad TARGCLIENT se establece en JMS en su lugar. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice true.

No
La propiedad TARGCLIENT del objeto Queue derivada del campo de cabecera JMSReplyTo de un mensaje de entrada siempre se establece en JMS.

Para los programas, utilice false.

TEMPMODEL

Nombre de la cola modelo a partir de la cual se crean colas temporales de JMS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPMODEL

Nombre abreviado de la herramienta de administración de JMS : TM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTemporaryModelo ()
- MQConnectionFactory.getTemporaryModelo ()

Valores

SYSTEM.DEFAULT.MODEL.QUEUE
Cualquier serie puede ser el valor predeterminado.

TEMPQPREFIX

Prefijo que se utiliza para formar el nombre de una cola dinámica de IBM MQ.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPQPREFIX

Nombre abreviado de la herramienta de administración de JMS : TQP

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory.getTempQPrefix ()

Valores

" " (serie vacía)
El prefijo utilizado es `CSQ.*` en z/OS y `AMQ.*` en todas las demás plataformas. Estos son los valores predeterminados.

Prefijo de cola
El prefijo de cola es cualquier serie que se ajuste a las reglas para formar contenido del campo *DynamicQName* en un descriptor de objeto IBM MQ (estructura MQOD), pero el último carácter no en blanco debe ser un asterisco.

TEMPTOPICPREFIX

Al crear temas temporales, JMS genera una serie de tema con el formato '' TEMP */TEMPTOPICPREFIX/ unique_id* '', o si esta propiedad se deja con el valor predeterminado, sólo '' TEMP */unique_id* ''. Especificar un TEMPTOPICPREFIX no vacío permite definir colas de modelo específicas para crear las colas gestionadas para suscriptores a temas temporales creados bajo esta conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPTOPICPREFIX

Nombre abreviado de la herramienta de administración de JMS : TTP

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Valores

Cualquier serie no nula que conste sólo de caracteres válidos para una serie de tema IBM MQ . El valor predeterminado es '' '' (serie vacía).

Tema

El nombre del destino de tema JMS , este valor lo utiliza el gestor de colas como serie de tema de una publicación o suscripción.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : TOPIC

Nombre abreviado de la herramienta de administración de JMS : TOP

Valores

Cualquier serie
Una serie que forma una serie de tema IBM MQ válida. Cuando utilice IBM MQ como proveedor de mensajería con WebSphere Application Server, especifique un valor que coincida con el nombre por el que se conoce el tema para fines administrativos en WebSphere Application Server.

Conceptos relacionados
Series de tema

TRANSPORT

La naturaleza de una conexión con un gestor de colas o intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : TRANSPORT

Nombre abreviado de la herramienta de administración de JMS : TRAN

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTransportTipo ()
- MQConnectionFactory.getTransportTipo ()

Valores

BIND
Para una conexión con un gestor de colas en modalidad de enlaces. Este es el valor predeterminado para las herramientas administrativas.

Para programas, utilice WMQConstants.WMQ_CM_BINDINGS.

CLIENT
Para una conexión con un gestor de colas en modalidad de cliente.

Para programas, utilice WMQConstants.WMQ_CM_CLIENT.

DIRECT
Para una conexión en tiempo real con un intermediario que no utiliza el túnel HTTP.

Para los programas, utilice WMQConstants.WMQ_CM_DIRECT_TCPIP.

DIRECTHTTP
Para una conexión en tiempo real a un intermediario utilizando el túnel HTTP. Solo se da soporte a HTTP 1.0 .

Para los programas, utilice WMQConstants.WMQ_CM_DIRECT_HTTP.

Conceptos relacionados

"Dependencias entre propiedades de objetos IBM MQ classes for JMS" en la página 1950
La validez de algunas propiedades depende de los valores particulares de otras propiedades.

WILDCARDFORMAT

Esta propiedad determina qué versión de sintaxis de comodín se va a utilizar.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : WILDCARDFORMAT

Nombre abreviado de la herramienta de administración de JMS : WCFMT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setWildCardFormat()
- MQConnectionFactory.getWildCardFormat()

Valores

TOPIC_ONLY
Sólo reconoce comodines de nivel de tema, tal como se utiliza en la versión 2 del intermediario. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_WILDCARD_TOPIC_ONLY.

CAR_SOLO
Sólo reconoce caracteres comodín, tal como se utiliza en la versión 1 del intermediario.

Para los programas, utilice WMQConstants.WMQ_WILDCARD_CHAR_ONLY.

La propiedad ENCODING

La propiedad ENCODING comprende tres subpropiedades, en doce combinaciones posibles.

Los valores válidos que puede tomar la propiedad ENCODING se construyen a partir de las tres subpropiedades:

Codificación de enteros
Normal o invertido

Codificación decimal
Normal o invertido

codificación de coma flotante
IEEE normal, IEEE invertido o z/OS

La propiedad ENCODING se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS
- El primer carácter representa *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad ENCODING .

Hay un valor adicional, la serie NATIVE, que establece los valores de codificación adecuados para la plataforma Java .

Los ejemplos siguientes muestran combinaciones válidas para ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

Propiedades TLS de los objetos JMS

Habilite el cifrado TLS (Transport Layer Security) utilizando la propiedad SSLCIPHERSUITE. A continuación, puede cambiar las características del cifrado TLS utilizando otras propiedades.

Cuando especifica TRANSPORT (CLIENT), puede habilitar la comunicación cifrada TLS utilizando la propiedad SSLCIPHERSUITE. Establezca esta propiedad en una CipherSuite válida proporcionada por el proveedor JSSE; debe coincidir con la CipherSpec especificada en el canal SVRCONN especificado por la propiedad CHANNEL.

Sin embargo, las CipherSpecs (tal como se especifica en el canal SVRCONN) y las CipherSuites (tal como se especifica en los objetos ConnectionFactory) utilizan distintos esquemas de denominación para representar los mismos algoritmos de cifrado TLS. Si se especifica un nombre CipherSpec reconocido en la propiedad SSLCIPHERSUITE, JMSAdmin emite un aviso y correlaciona la CipherSpec con su CipherSuiteequivalente. Consulte TLS CipherSpecs y CipherSuites en IBM MQ classes for JMS para obtener una lista de CipherSpecs reconocidas por IBM MQ y JMSAdmin.

Si necesita una conexión para utilizar una CipherSuite que esté soportada por el proveedor de IBM Java JSSE FIPS (IBMJSSEFIPS), establezca la propiedad SSLFIPSREQUIRED de la fábrica de conexiones en YES. El valor predeterminado de esta propiedad es NO, lo que significa que una conexión puede utilizar cualquier CipherSuitesoportada. La propiedad se ignora si no se ha establecido SSLCIPHERSUITE.

El SSLPEERNAME coincide con el formato del parámetro SSLPEER, que se puede establecer en definiciones de canal. Es una lista de pares de nombre-valor de atributo separados por comas o signos de punto y coma. Por ejemplo:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

El conjunto de nombres y valores forma un *nombre distinguido*. Para obtener más detalles sobre los nombres distinguidos y su uso con IBM MQ, consulte Protección IBM MQ.

El ejemplo proporcionado comprueba el certificado de identificación presentado por el servidor durante la conexión. Para que la conexión sea satisfactoria, el certificado debe tener un nombre común que empiece por QMGR., y debe tener al menos dos nombres de unidad organizativa, el primero de los cuales es IBM y el segundo WEBSPHERE. La comprobación no distingue entre mayúsculas y minúsculas.

Si SSLPEERNAME no está establecido, no se realiza ninguna comprobación de este tipo. SSLPEERNAME se ignora si no se ha establecido SSLCIPHERSUITE.

La propiedad SSLCRL especifica cero o más servidores CRL (Lista de revocación de certificados). El uso de esta propiedad requiere una JVM en Java 2 v1.4. Esta es una lista delimitada por espacios de entradas con el formato:

```
ldap:// hostname:[ port ]
```

seguido opcionalmente por un único/. Si se omite *port* , se presupone el puerto LDAP predeterminado de 389. En el momento de la conexión, el certificado TLS presentado por el servidor se compara con los servidores CRL especificados. Consulte Protección IBM MQ para obtener más información sobre la seguridad de CRL.

Si SSLCRL no está establecido, no se realiza ninguna comprobación de este tipo. SSLCRL se ignora si no se ha establecido SSLCIPHERSUITE.

La propiedad SSLRESETCOUNT representa el número total de bytes enviados y recibidos por una conexión antes de que se renegocie la clave secreta que se utiliza para el cifrado. El número de bytes enviados es el número antes del cifrado y el número de bytes recibidos es el número después del cifrado. El número de bytes también incluye información de control enviada y recibida por IBM MQ classes for JMS.

Por ejemplo, para configurar un objeto ConnectionFactory que se puede utilizar para crear una conexión sobre un canal MQI habilitado para TLS, con una clave secreta que se renegocia después de que se hayan enviado 4 MB de datos, emita el siguiente mandato a JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Si el valor de SSLRESETCOUNT es cero, que es el valor predeterminado, la clave secreta nunca se renegocia. La propiedad SSLRESETCOUNT se ignora si SSLCIPHERSUITE no está establecido.

Referencia de IBM Message Service Client for .NET

Esta sección de referencia proporciona información sobre las interfaces de clase IBM Message Service Client for .NET (XMS .NET) y sobre las propiedades de objeto definidas por XMS.

.NET interfaces

En esta sección se describen las interfaces de clase .NET y sus propiedades y métodos.

En la tabla siguiente se resumen las interfaces, que están definidas en el espacio de nombres `IBM.XMS` .

Tabla 871. Resumen de las interfaces de clase .NET	
Interface	**Descripción**
“IBytesMessage” en la página 2005	Un mensaje de bytes es un mensaje cuyo cuerpo está formado por una corriente de bytes.
“IConnection” en la página 2015	Un objeto Connection representa la conexión activa de la aplicación a un servidor de mensajería.
“IConnectionFactory” en la página 2017	Una aplicación utiliza una fábrica de conexiones para crear una conexión.
“IConnectionMetaData” en la página 2019	Un objeto ConnectionMetaData proporciona información sobre una conexión.
“IDestination” en la página 2019	Un destino es el lugar al que una aplicación envía mensajes, o es un origen del cual una aplicación recibe mensajes, o ambas cosas.
“ExceptionListener” en la página 2020	Una aplicación utiliza un escucha de excepción al que se le notificará de forma asíncrona un problema con una conexión.
“IllegalStateException” en la página 2021	XMS emite esta excepción si una aplicación llama a un método en un momento incorrecto o inapropiado, o si XMS no está en un estado adecuado para la operación solicitada.
“InitialContext” en la página 2021	Una aplicación utiliza un objeto InitialContext para crear objetos a partir de definiciones de objeto que se han recuperado de un repositorio de objetos administrados.
“InvalidClientIDException” en la página 2023	XMS emite esta excepción si una aplicación intenta establecer un identificador de cliente para una conexión, pero el identificador de cliente no es válido o ya está en uso.
“InvalidDestinationException” en la página 2024	XMS emite esta excepción si una aplicación especifica un destino que no es válido.
“InvalidSelectorException” en la página 2024	XMS emite esta excepción si una aplicación proporciona una expresión de selector de mensajes cuya sintaxis no es válida.
“IMapMessage” en la página 2024	Un mensaje de correlación es un mensaje cuyo cuerpo está formado por un conjunto de pares de nombre-valor, donde cada valor tiene un tipo de datos asociado.
“IMessage” en la página 2033	Un objeto Message representa un mensaje que una aplicación envía o recibe. IMessage es una superclase para las clases de mensaje como, por ejemplo, IMapMessage.
“IMessageConsumer” en la página 2039	Una aplicación utiliza un consumidor de mensaje para recibir mensajes enviados a un destino.
“MessageEOFException” en la página 2041	XMS emite esta excepción si XMS encuentra el final de una corriente de mensajes de bytes cuando una aplicación está leyendo el cuerpo de un mensaje de bytes.

Tabla 871. Resumen de las interfaces de clase .NET (continuación)	
Interface	**Descripción**
“MessageFormatException” en la página 2042	XMS emite esta excepción si XMS encuentra un mensaje con un formato que no es válido.
“IMessageListener (delegado)” en la página 2042	Una aplicación utiliza un escucha de mensajes para recibir mensajes de forma asíncrona.
“MessageNotReadableException” en la página 2042	XMS emite esta excepción si una aplicación intenta leer el cuerpo de un mensaje que es sólo de escritura.
“MessageNotWritableException” en la página 2043	XMS emite esta excepción si una aplicación intenta grabar en el cuerpo de un mensaje que es de sólo lectura.
“IMessageProducer” en la página 2043	Una aplicación utiliza un productor de mensajes para enviar mensajes a un destino.
“IObjectMessage” en la página 2048	Un mensaje de objeto es un mensaje cuyo cuerpo está formado por un objeto Java o .NET serializado.
“IPropertyContext” en la página 2049	IPropertyContext es una superclase abstracta que contiene métodos que obtienen y establecen propiedades. Estos métodos son heredados por otras clases.
“IQueueBrowser” en la página 2058	Una aplicación utiliza un examinador de colas para examinar mensajes en una cola sin eliminarlas.
“Solicitante” en la página 2060	Una aplicación utiliza un solicitante para enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla.
“ResourceAllocationException” en la página 2061	XMS emite esta excepción si XMS no puede asignar los recursos necesarios para un método.
“SecurityException” en la página 2061	XMS emite esta excepción si se rechazan el identificador de usuario y la contraseña proporcionados para autenticar una aplicación. XMS también lanza esta excepción si una comprobación de autoridad falla e impide que se complete un método.
“ISession” en la página 2062	Una sesión es un contexto de hebra única para enviar y recibir mensajes.
“IStreamMessage” en la página 2072	Un mensaje de corriente de datos es un mensaje cuyo cuerpo está formado por una corriente de valores, donde cada valor tiene un tipo de datos asociado.
“ITextMessage” en la página 2081	Un mensaje de texto es un mensaje cuyo cuerpo está formado por una serie.
“TransactionInProgressException” en la página 2082	XMS emite esta excepción si una aplicación solicita una operación que no es válida porque hay una transacción en curso.
“TransactionRolledBackException” en la página 2082	XMS emite esta excepción si una aplicación llama a Session.commit() para confirmar la transacción actual, pero la transacción se retrotrae.

Tabla 871. Resumen de las interfaces de clase .NET (continuación)	
Interface	**Descripción**
XMSC	Para .NET, los valores y nombres de propiedad XMS se definen en esta clase como constantes públicas. Si desea más detalles, consulte “Propiedades de objetos XMS” en la página 2085.
“XMSException” en la página 2082	Si XMS detecta un error al procesar una llamada a un método .NET , XMS emite una excepción. Una excepción es un objeto que encapsula información sobre el error. Hay distintos tipos de excepción XMS y un objeto XMSException es sólo un tipo de excepción. Sin embargo, la clase XMSException es una superclase de las otras clases de excepción XMS. XMS lanza un objeto XMSException en situaciones donde ninguno de los otros tipos de excepción es apropiado.
“XMSFactoryFactory” en la página 2083	Si una aplicación no está utilizando objetos administrados, utilice esta clase para crear fábricas de conexiones, colas y temas.

La definición de cada método lista los códigos de excepción que podría devolver XMS si detecta un error al procesar una llamada al método. Cada código de excepción se representa mediante su constante con nombre, que tiene una excepción correspondiente.

IBytesMessage

Un mensaje de bytes es un mensaje cuyo cuerpo está formado por una corriente de bytes.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

Propiedades de .NET

BodyLength - Obtener longitud de cuerpo

Interfaz:

```
Int64 BodyLength
    {
      get;
    }
```

Obtener la longitud del cuerpo del mensaje en bytes cuando el cuerpo del mensaje es de solo lectura.

El valor devuelto es la longitud de todo el cuerpo independientemente de la posición actual del cursor para leer el mensaje.

Excepciones:

- XMSException
- MessageNotReadableException

Métodos

ReadBoolean - Leer valor booleano

Interfaz:

```
Boolean ReadBoolean();
```

Leer un valor booleano de la corriente de datos del mensaje de bytes.

Parámetros:
Ninguna

Devuelve:
El valor booleano que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - Leer byte

Interfaz:

```
Int16   ReadSignedByte();
```

Leer el siguiente byte de la corriente de datos del mensaje de bytes como un entero de 8-bits firmado.

Parámetros:
Ninguna

Devuelve:
El byte que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Leer bytes

Interfaz:

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

Leer una matriz de bytes de la corriente de datos del mensaje de bytes empezando por la posición actual del cursor.

Parámetros:

array (salida)
El almacenamiento intermedio que va a contener la matriz de bytes que se ha leído. Si el número de bytes que faltan por leer de la corriente de datos antes de la llamada es mayor o igual que la longitud del almacenamiento intermedio, el almacenamiento intermedio se rellena. De lo contrario, el almacenamiento intermedio se rellena de forma parcial con todos los bytes restantes.

Si especifica un puntero nulo en la entrada, el método se salta los bytes sin leerlos. Si el número de bytes que faltan por leer de la corriente de datos antes de la llamada es mayor o igual que la longitud del almacenamiento intermedio, el número de bytes omitidos es igual a la longitud

del almacenamiento intermedio. De lo contrario, se omiten todos los bytes restantes. El cursor permanece en la siguiente posición para leer en la corriente de datos del mensaje de byte.

length (entrada)
La longitud del almacenamiento intermedio en bytes

Devuelve:
El número de bytes que se van a leer en el almacenamiento intermedio. Si el almacenamiento intermedio se rellena de forma parcial, el valor es menor que la longitud del almacenamiento intermedio, que indica que no queda ningún byte más para leer. Si no queda ningún bytes para leer en la corriente de datos antes de la llamada, el valor es `XMSC_END_OF_STREAM`.

Si especifica un puntero nulo en la entrada, el método no devuelve ningún valor.

Excepciones:

- XMSException
- MessageNotReadableException

ReadChar - Leer carácter

Interfaz:

```
Char    ReadChar();
```

Leer los 2 bytes siguientes de la corriente de datos del mensaje de bytes como un carácter.

Parámetros:
Ninguna

Devuelve:
El carácter que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Leer número de coma flotante de precisión doble

Interfaz:

```
Double  ReadDouble();
```

Leer los 8 bytes siguiente de la corriente de datos del mensaje de bytes como un número de coma flotante de precisión doble.

Parámetros:
Ninguna

Devuelve:
El número de coma flotante de precisión doble que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leer número de coma flotante

Interfaz:

```
Single  ReadFloat();
```

Leer los 4 bytes siguientes de la corriente de datos del mensaje de bytes como un número de coma flotante.

Parámetros:
Ninguna

Devuelve:
El número de coma flotante que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Leer entero

Interfaz:

```
Int32   ReadInt();
```

Leer los 4 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 32-bits firmado.

Parámetros:
Ninguna

Devuelve:
El entero que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Leer entero largo

Interfaz:

```
Int64   ReadLong();
```

Leer los 8 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 64-bits firmado.

Parámetros:
Ninguna

Devuelve:
El entero largo que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadShort - Leer entero corto

Interfaz:

```
Int16   ReadShort();
```

Leer los 2 bytes siguientes de la corriente de datos del mensaje de bytes como un entero de 16-bits firmado.

Parámetros:
Ninguna

Devuelve:
El entero corto que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Leer byte sin firmar

Interfaz:

```
Byte    ReadByte();
```

Leer el siguiente byte de la corriente de datos del mensaje de bytes como un entero de 8-bits sin firmar.

Parámetros:
Ninguna

Devuelve:
El byte que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - Leer entero corto sin firmar

Interfaz:

```
Int32   ReadUnsignedShort();
```

Leer los 2 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 16-bits sin firmar.

Parámetros:
Ninguna

Devuelve:
El entero corto sin firma que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF - Leer serie UTF

Interfaz:

```
String  ReadUTF();
```

Leer una serie, codificada en UTF-8, de la corriente de datos del mensaje de bytes.

Nota: Antes de llamar a ReadUTF(), asegúrese de que el cursor del almacenamiento intermedio está apuntado al inicio de la corriente de datos del mensaje de bytes.

Parámetros:
Ninguna

Devuelve:
Un objeto String que encapsula la serie que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Restablecer

Interfaz:

```
void Reset();
```

Colocar el cuerpo del mensaje en la modalidad de solo lectura y volver a colocar el cursor al inicio de la corriente de datos del mensaje de bytes.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotReadableException

WriteBoolean - Escribir valor booleano

Interfaz:

```
void  WriteBoolean(Boolean value);
```

Escribir un valor booleano en la corriente de datos de mensaje de bytes.

Parámetros:

value (entrada)
El valor booleano que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteByte - Escribir byte

Interfaz:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Escribir un byte en la corriente de datos del mensaje de bytes.

Parámetros:

value (entrada)
El byte que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteBytes - Escribir bytes

Interfaz:

```
void  WriteBytes(Byte[] value);
```

Escribir una matriz de bytes en la corriente de datos del mensaje de bytes.

Parámetros:

value (entrada)
La matriz de bytes que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteBytes - Escribir matriz de bytes parcial

Interfaz:

```
void  WriteBytes(Byte[] value, int offset, int length);
```

Escribir una matriz de bytes parcial en la corriente de datos del mensaje de bytes, tal como lo ha definido la longitud especificada.

Parámetros:

value (entrada)
La matriz de bytes que se va a escribir.

offset (entrada)
El punto de inicio para la matriz de bytes que se va a escribir.

length (entrada)
El número de bytes que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteChar - Escribir carácter

Interfaz:

```
void  WriteChar(Char value);
```

Escribir un carácter en la corriente de datos del mensaje de bytes como 2 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El carácter que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteDouble - Escribir número de coma flotante de precisión doble

Interfaz:

```
void  WriteDouble(Double value);
```

Convertir un número de coma flotante de precisión doble a un entero largo y escribir el entero largo en la corriente de datos del mensaje de bytes como 8 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El número de coma flotante de precisión doble que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteFloat - Escribir número de coma flotante

Interfaz:

```
void  WriteFloat(Single value);
```

Convertir un número de coma flotante a un entero y escribir el entero en la corriente de datos del mensaje de bytes como 4 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El número de coma flotante que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteInt - Escribir entero

Interfaz:

```
void  WriteInt(Int32 value);
```

Escribir un entero en la corriente de datos del mensaje de bytes como 4 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteLong - Escribir entero largo

Interfaz:

```
void  WriteLong(Int64 value);
```

Escribir un entero largo en la corriente de datos del mensaje de bytes como 8 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero largo que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteObject - Escribir objeto

Interfaz:

```
void  WriteObject(Object value);
```

Escribir el objeto especificado en la corriente de datos del mensaje de bytes.

Parámetros:

value (entrada)
El objeto que se va a escribir, que debe hacer referencia a u n tipo primitivo.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteShort - Escribir entero corto

Interfaz:

```
void  WriteShort(Int16 value);
```

Escribir un entero corto en la corriente de datos del mensaje de bytes como 2 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero corto que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteUTF - Escribir serie UTF

Interfaz:

```
void  WriteUTF(String value);
```

Escribir una serie, codificada en UTF-8, en la corriente de datos del mensaje de bytes.

Parámetros:

value (entrada)
Un objeto String que encapsula la serie que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody, clearProperties, PropertyExists

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnection

Un objeto Connection representa la conexión activa de la aplicación a un servidor de mensajería.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

Si desea una lista de las propiedades definidas de XMS de un objeto Connection, consulte “Propiedades de conexión” en la página 2086.

Propiedades de .NET

ClientID - Obtener y establecer ID de cliente

Interfaz:

```
String ClientID
      {
        get;
        set;
      }
```

Obtener y establecer el identificador de cliente para la conexión.

El administrador puede preconfigurar el identificador de cliente en objeto ConnectionFactory, o asignarlo estableciendo ClientID.

Un identificador de cliente se utiliza solo para dar soporte a suscripciones duraderas en el dominio de publicación/suscripción y se ignora en el dominio punto a punto.

Si una aplicación establece un identificador de cliente para una conexión, la aplicación debe hacerlo inmediatamente después de crear la conexión y antes de realizar cualquier otra operación en la conexión. Si la aplicación intenta establecer un identificador de cliente después de este punto, la llamada lanza la excepción IllegalStateException.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

Excepciones:

- XMSException
- IllegalStateException
- InvalidClientIDException

ExceptionListener - Obtener y establecer escucha de excepción

Interfaz:

```
    ExceptionListener ExceptionListener
        {
          get;
          set;
        }
```

Obtener el escucha de excepción que está registro con la conexión y registrar un escucha de excepción con la conexión.

Si no hay registrado ningún escucha de excepción con la conexión, el método devuelve un valor nulo. Si ya hay un escucha de excepción registrado con la conexión, puede cancelar el registro especificando un valor nulo, en lugar del escucha de excepción.

Para obtener más información sobre cómo utilizar escuchas de excepciones, consulte Utilización de escuchas de mensajes y excepciones en .NET.

Excepciones:

- XMSException

Metadata - Obtener metadatos

Interfaz:

```
IConnectionMetaData MetaData
    {
      get;
    }
```

Obtener los metadatos para la conexión.

Excepciones:

- XMSException

Métodos

Close - Cerrar conexión

Interfaz:

```
void  Close();
```

Cerrar la conexión.

Si una aplicación intenta cerrar una conexión que ya está cerrar, la llamada se ignora.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

CreateSession - Crear sesión

Interfaz:

```
    ISession CreateSession(Boolean transacted,
                           AcknowledgeMode acknowledgeMode);
```

Crear una sesión.

Parámetros:

transacted (entrada)
El valor `True` significa que la sesión es una sesión con transacción. El valor `False` significa que la sesión no es una sesión con transacción.

Para una conexión en tiempo real con un intermediario, el valor debe ser `False`.

acknowledgeMode (entrada)
Indica cómo se acusa el recibo de los mensajes recibidos por una aplicación. El valor debe adoptar uno de los valores siguientes del enumerador AcknowledgeMode:

`AcknowledgeMode.AutoAcknowledge`

`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Para una conexión en tiempo real con un intermediario, el valor debe ser `AcknowledgeMode.AutoAcknowledge` o `AcknowledgeMode.DupsOkAcknowledge`

Este parámetro se ignora si la sesión es una sesión con transacción. Para obtener más información sobre las modalidades de acuse de recibo, consulte Acuse de recibo de mensaje.

Devuelve:
El objeto Session

Excepciones:

- XMSException

Start - Iniciar conexión

Interfaz:

```
void  Start();
```

Iniciar o reiniciar la entrega de mensajes entrantes para la conexión. La llamada se ignora si la conexión ya se ha iniciado.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

Stop - Detener conexión

Interfaz:

```
void  Stop();
```

Detener la entrega de mensajes entrantes para la conexión. La llamada se ignora, si la conexión ya se ha detenido.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionFactory

Una aplicación utiliza una fábrica de conexiones para crear una conexión.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

Si desea una lista de las propiedades definidas de XMS de un objeto ConnectionFactory, consulte “Propiedades de ConnectionFactory” en la página 2086.

Métodos

CreateConnection - Crear fábrica de conexiones (utilizando la identidad de usuario predeterminada)

Interfaz:

```
IConnection CreateConnection();
```

Crear una fábrica de conexiones con las propiedades predeterminadas.

Si se está conectando a IBM MQ y XMSC_USERID no está establecida, el gestor de colas utiliza el ID de usuario del usuario conectado, de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ.

Parámetros:
Ninguna

Excepciones:

- XMSException

CreateConnection - Crear conexión (utilizando una identidad de usuario especificada)

Interfaz:

```
IConnection CreateConnection(String userId, String password);
```

Crear una conexión utilizando una identidad de usuario especificada.

Si se está conectando a IBM MQ y XMSC_USERID no está establecida, el gestor de colas utiliza el ID de usuario del usuario conectado, de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ.

La conexión se crea en la modalidad detenida. No se entrega ningún mensaje hasta que la aplicación llama a **`Connection.start()`**.

Parámetros:

userID (entrada)
Un objeto String que encapsula el identificador de usuario que se va a utilizar para autenticar la aplicación. Si proporciona un valor nulo, se realiza un intento de crear la conexión sin autenticación.

password (entrada)
Un objeto String que encapsula la contraseña que se va a utilizar para autenticar la aplicación. Si proporciona un valor nulo, se realiza un intento de crear la conexión sin autenticación.

Devuelve:
El objeto Connection.

Excepciones:

- XMSException

- XMS_X_SECURITY_EXCEPTION

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionMetaData

Un objeto ConnectionMetaData proporciona información sobre una conexión.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

Si desea una lista de las propiedades definidas de XMS de un objeto ConnectionMetaData, consulte “Propiedades de ConnectionMetaData” en la página 2092.

Propiedades de .NET

JMSXPropertyNames - Obtener propiedades de mensaje definidas de JMS

Interfaz:

```
System.Collections.IEnumerator JMSXPropertyNames
    {
      get;
    }
```

Devolver una enumeración de los nombres de las propiedades de mensaje definidas por JMS soportadas por la conexión.

Las propiedades de mensaje definidas de JMS no están soportadas por una conexión en tiempo real con un intermediario.

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IDestination

Un destino es el lugar al que una aplicación envía mensajes, o es un origen del cual una aplicación recibe mensajes, o ambas cosas.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

Si desea una lista de las propiedades definidas de XMS de un objeto Destination, consulte “Propiedades de Destination” en la página 2092.

Propiedades de .NET

Name - Obtener nombre de destino

Interfaz:

```
String Name
{
  get;
}
```

Obtener el nombre del destino. El nombre es una serie que encapsula el nombre de una cola, o bien el nombre de un tema.

Excepciones:

- XMSException

TypeId - Obtener tipo de destino

Interfaz:

```
DestinationType TypeId
{
  get;
}
```

Obtener el tipo del destino. El tipo del destino es uno de los valores siguientes:

`DestinationType.Queue`
`DestinationType.Topic`

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ExceptionListener

Una aplicación utiliza un escucha de excepción al que se le notificará de forma asíncrona un problema con una conexión.

Jerarquía de herencia:
Ninguna

Si una aplicación utiliza una conexión solo para consumir mensajes de forma asíncrona, y sin ninguna otra finalidad, la única forma a través de la cual la aplicación puede obtener información sobre un problema con la conexión es utilizar un escucha de excepción. En otras situaciones, un escucha de excepción puede proporcionar una forma más inmediata de obtener información sobre un problema con una conexión que esperar hasta la siguiente llamada síncrona a XMS.

Delegado

ExceptionListener - Escucha de excepción

Interfaz:

```
public delegate void ExceptionListener(Exception ex)
```

Notifique a la aplicación un problema con una conexión.

Los métodos que implementan este delegado se pueden registrar con la conexión.

Para obtener más información sobre cómo utilizar escuchas de excepciones, consulte Utilización de escuchas de mensajes y excepciones en .NET.

Parámetros:

exception (entrada)
Un puntero para una excepción creada por XMS.

Devuelve:
Void

IllegalStateException

XMS emite esta excepción si una aplicación llama a un método en un momento incorrecto o inapropiado, o si XMS no está en un estado adecuado para la operación solicitada.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
           |
           +----IBM.XMS.IllegalStateException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

InitialContext

Una aplicación utiliza un objeto InitialContext para crear objetos a partir de definiciones de objeto que se han recuperado de un repositorio de objetos administrados.

Jerarquía de herencia:
Ninguna

Propiedades de .NET

Environment - Obtener el entorno

Interfaz:

```
Hashtable Environment
```

```
{
     get;
}
```

Obtener el entorno.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Constructores

InitialContext - Crear contexto inicial

Interfaz:

```
InitialContext(Hashtable env);
```

Crear un objeto InitialContext.

Parámetros:

La información necesaria para establecer una conexión al repositorio de objetos administrados se proporciona al constructor en una tabla hash de entorno.

Excepciones:

- XMSException

Métodos

AddToEnvironment - Añadir una nueva propiedad al entorno

Interfaz:

```
Object AddToEnvironment(String propName, Object propVal);
```

Añadir una nueva propiedad al entorno.

Parámetros:

propName (entrada
Un objeto String que encapsula el nombre de la propiedad que se va a añadir.

propVal (entrada)
El valor de la propiedad que se va añadir.

Devuelve:
El valor antiguo de la propiedad.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Close - Cerrar este contexto

Interfaz:

```
void Close()
```

Cerrar este contexto.

Parámetros:
Ninguna

Devuelve:
Ninguna

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Lookup - Buscar objeto en contexto inicial

Interfaz:

```
Object Lookup(String name);
```

Crear un objeto a partir de una definición de objeto que se ha recuperado del repositorio de objetos administrados.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre del objeto administrado que se va a recuperar. El nombre puede ser un nombre sencillo o un nombre complejo. Para obtener más detalles, consulte Recuperación de objetos administrados.

Devuelve:
IConnectionFactory o IDestination, en función del tipo de objeto que se está recuperando. Si la función puede acceder al directorio, pero no puede encontrar el objeto necesario, se devuelve un valor nulo.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

RemoveFromEnvironment - Eliminar una propiedad del entorno

Interfaz:

```
Object RemoveFromEnvironment(String propName);
```

Eliminar una propiedad del entorno.

Parámetros:

propName (entrada
Un objeto String que encapsula el nombre de la propiedad que se va a eliminar.

Devuelve:
El objeto que se ha eliminado.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

InvalidClientIDException

XMS emite esta excepción si una aplicación intenta establecer un identificador de cliente para una conexión, pero el identificador de cliente no es válido o ya está en uso.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

InvalidDestinationException

XMS emite esta excepción si una aplicación especifica un destino que no es válido.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

InvalidSelectorException

XMS emite esta excepción si una aplicación proporciona una expresión de selector de mensajes cuya sintaxis no es válida.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidSelectorException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

IMapMessage

Un mensaje de correlación es un mensaje cuyo cuerpo está formado por un conjunto de pares de nombre-valor, donde cada valor tiene un tipo de datos asociado.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

Cuando una aplicación obtiene el valor del par nombre-valor, el valor se puede convertir mediante XMS a otro tipo de datos. Para obtener más información sobre esta forma de conversión implícita, consulte la información sobre los mensajes de correlación en El cuerpo de un mensaje XMS.

Propiedades de .NET

MapNames - Obtener nombres de correlación

Interfaz:

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

Obtener una enumeración de los nombres en el cuerpo del mensaje de correlación.

Excepciones:

- XMSException

Métodos

GetBoolean - Obtener valor booleano

Interfaz:

```
Boolean GetBoolean(String name);
```

Obtener el valor booleano identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el valor booleano.

Devuelve:
El valor booleano recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetByte - Obtener byte

Interfaz:

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

Obtener el byte identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el byte.

Devuelve:
El byte recuperado del cuerpo del mensaje de correlación. No se realiza ninguna conversión de datos en el byte.

Excepciones:

- XMSException

GetBytes - Obtener bytes

Interfaz:

```
Byte[]  GetBytes(String name);
```

Obtener la matriz de bytes identificada por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica la matriz de bytes.

Devuelve:
El número de bytes de la matriz.

Excepciones:

- XMSException

GetChar - Obtener carácter

Interfaz:

```
Char    GetChar(String name);
```

Obtener el carácter identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el carácter.

Devuelve:
El carácter recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetDouble - Obtener número de coma flotante de precisión doble

Interfaz:

```
Double  GetDouble(String name);
```

Obtener el número de coma flotante de precisión doble identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el número de coma flotante de precisión doble.

Devuelve:
El número de coma flotante de precisión doble recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetFloat - Obtener número de coma flotante

Interfaz:

```
Single  GetFloat(String name);
```

Obtener el número de coma flotante identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el número de coma flotante.

Devuelve:
El número de coma flotante recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetInt - Obtener entero

Interfaz:

```
Int32   GetInt(String name);
```

Obtener el entero identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el entero.

Devuelve:
El entero recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetLong - Obtener entero largo

Interfaz:

```
Int64   GetLong(String name);
```

Obtener el entero largo identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el entero largo.

Devuelve:
El entero largo recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetObject - Obtener objeto

Interfaz:

```
Object  GetObject(String name);
```

Obtener una referencia al valor de un par de nombre-valor del cuerpo del mensaje de correlación. El par nombre-valor se identifica por el nombre.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre del par nombre-valor.

Devuelve:
El valor, que es uno de los tipos de objeto siguientes:

```
Boolean
Byte
Byte[]
```

```
Char
Double
Single
Int32
Int64
Int16
String
```

Excepciones:
XMSException

GetShort - Obtener entero corto

Interfaz:

```
Int16   GetShort(String name);
```

Obtener el entero corto identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica el entero corto.

Devuelve:
El entero corto recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSException

GetString - Obtener serie

Interfaz:

```
String  GetString(String name);
```

Obtener la serie identificada por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre que identifica la serie en el cuerpo del mensaje de correlación.

Devuelve:
Un objeto String que encapsula la serie recuperada del cuerpo del mensaje de correlación. Si es necesaria una conversión de datos, este valor es la serie después de la conversión.

Excepciones:

- XMSException

ItemExists - Comprobar si existe el par nombre-valor

Interfaz:

```
Boolean ItemExists(String name);
```

Comprobar si el cuerpo del mensaje de correlación contiene un par nombre-valor con el nombre especificado.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre del par nombre-valor.

Devuelve:

- `True`, si el cuerpo del mensaje de correlación contiene un par nombre-valor con el nombre especificado.
- `False`, si el cuerpo del mensaje de correlación no contiene un par nombre-valor con el nombre especificado.

Excepciones:

- XMSException

SetBoolean - Establecer valor booleano

Interfaz:

```
void  SetBoolean(String name, Boolean value);
```

Establecer un valor booleano en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el valor booleano en el cuerpo del mensaje de correlación.

value (entrada)
El valor booleano que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetByte - Establecer byte

Interfaz:

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

Establecer un byte en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el byte en el cuerpo del mensaje de correlación.

value (entrada)
El byte que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetBytes - Establecer bytes

Interfaz:

```
void  SetBytes(String name, Byte[] value);
```

Establecer una matriz debytes en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar la matriz de bytes en el cuerpo del mensaje de correlación.

value (entrada)
La matriz de bytes que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetChar - Establecer carácter

Interfaz:

```
void  SetChar(String name, Char value);
```

Establecer un carácter de 2-bytes en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el carácter en el cuerpo del mensaje de correlación.

value (entrada)
El carácter que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetDouble - Establecer número de coma flotante de precisión doble

Interfaz:

```
void  SetDouble(String name, Double value);
```

Establecer un número de coma flotante de precisión doble en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el número de coma flotante de precisión doble en el cuerpo del mensaje de correlación.

value (entrada)
El número de coma flotante de precisión doble que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetFloat - Establecer número de coma flotante

Interfaz:

```
void  SetFloat(String name, Single value);
```

Establecer un número de coma flotante en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el número de coma flotante en el cuerpo del mensaje de correlación.

value (entrada)
El número de coma flotante que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetInt - Establecer entero

Interfaz:

```
void  SetInt(String name, Int32 value);
```

Establecer un entero en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el entero en el cuerpo del mensaje de correlación.

value (entrada)
El entero que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetLong - Establecer entero largo

Interfaz:

```
void  SetLong(String name, Int64 value);
```

Establecer un entero largo en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el entero largo en el cuerpo del mensaje de correlación.

value (entrada)
El entero largo que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetObject - Establecer objeto

Interfaz:

```
void  SetObject(String name, Object value);
```

Establecer un valor, que debe ser un tipo primitivo XMS, en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el valor en el cuerpo del mensaje de correlación.

value (entrada)
Una matriz de bytes que contiene el valor que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetShort - Establecer entero corto

Interfaz:

```
void  SetShort(String name, Int16 value);
```

Establecer un entero corto en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar el entero corto en el cuerpo del mensaje de correlación.

value (entrada)
El entero corto que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

SetString - Establecer serie

Interfaz:

```
void  SetString(String name, String value);
```

Establecer una serie en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre para identificar la serie en el cuerpo del mensaje de correlación.

value (entrada)
Un objeto String que encapsula la serie que se va a establecer.

Devuelve:
Void

Excepciones:

- XMSException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody, clearProperties, PropertyExists

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessage

Un objeto Message representa un mensaje que una aplicación envía o recibe. IMessage es una superclase para las clases de mensaje como, por ejemplo, IMapMessage.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

Para obtener una lista de los campos de cabecera de mensaje JMS en un objeto Message, consulte Campos de cabecera de un mensaje XMS. Para obtener una lista de las propiedades definidas por JMS de un objeto Message, consulte Propiedades definidas por JMS de un mensaje. Para obtener una lista de las propiedades definidas de IBM de un objeto Message, consulte Propiedades definidas porIBMde un mensaje. Para obtener una lista de las propiedades JMS_IBM_MQMD* para el objeto Message, consulte “Propiedades JMS_IBM_MQMD*” en la página 2096

Los mensajes son suprimidos por el recopilador de basura. Cuando se suprime un mensaje, esta acción libera los recursos que estaba utilizando.

Propiedades de .NET

GetJMSCorrelationID - Obtener y establecer JMSCorrelationID

Interfaz:

```
String JMSCorrelationID
{
  get;
  set;
}
```

Obtener y establecer el identificador de correlación del mensaje como un objeto String.

Excepciones:

- XMSException

JMSDeliveryMode - Obtener y establecer JMSDeliveryMode

Interfaz:

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

Obtener y establecer la modalidad de entrega del mensaje.

La modalidad de entrega del mensaje adopta uno de los valores siguientes:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Para un mensaje recién creado que no se ha enviado, la modalidad de entrega es `DeliveryMode.Persistent`, excepto para una conexión en tiempo real a un intermediario para el cual la modalidad de entrega es `DeliveryMode.NonPersistent`. Para un mensaje que se recibe, el método devuelve la modalidad de entrega que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie la modalidad de entrega estableciendo JMSDeliveryMode.

Excepciones:

- XMSException

JMSDestination - Obtener y establecer JMSDestination

Interfaz:

```
IDestination JMSDestination
{
  get;
  set;
}
```

Obtener y establecer el destino del mensaje.

El destino se establece mediante la llamada IMessageProducer.send() cuando se envía el mensaje. Se ignora el valor de JMSDestination. Sin embargo, puede utilizar JMSDestination para cambiar el destino de un mensaje que se ha recibido.

Para un mensaje recién creado que no se ha enviado, el método devuelve un objeto Destination nulo, a menos que la aplicación emisora establezca un destino estableciendo JMSDestination. Para un mensaje que se ha recibido, el método devuelve un objeto Destination para el destino que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie el destino estableciendo JMSDestination.

Excepciones:

- XMSException

JMSExpiration - Obtener y establecer JMSExpiration

Interfaz:

```
Int64 JMSExpiration
{
  get;
```

```
  set;
}
```

Obtener y establecer la hora de caducidad del mensaje.

La hora de caducidad se establece mediante la llamada IMessageProducer.send() cuando se envía el mensaje. Su valor se calcula añadiendo el tiempo de vida, tal como lo especifica la aplicación emisora, a la hora cuando se envía el mensaje. La hora de caducidad se expresa en milisegundos desde las 00:00:00 GMT el 1 de enero de 1970.

Para un mensaje recién creado que no se ha enviado, la hora de caducidad es 0, a menos que la aplicación emisora defina una hora de caducidad diferente estableciendo JMSExpiration. Para un mensaje que se ha recibido, el método devuelve la hora caducidad que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie la hora de caducidad estableciendo JMSExpiration.

Si la hora de caducidad es 0, la llamada IMessageProducer.send() establece la hora de caducidad en 0 para indicar que el mensaje no caduca.

XMS descarta mensajes caducados y no los entrega a aplicaciones.

Excepciones:

- XMSException

JMSMessageID - Obtener y establecer JMSMessageID

Interfaz:

```
String JMSMessageID
{
  get;
  set;
}
```

Obtener y establecer el identificador de mensaje del mensaje como un objeto de serie que encapsula el identificador de mensaje.

El identificador de mensaje se establece mediante la llamada IMessageProducer.send() cuando se envía el mensaje. Para un mensaje que se ha recibido, el método devuelve el identificador de mensaje que estableció la llamadaIMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie el identificador de mensaje estableciendo JMSMessageID.

Si el mensaje no tiene ningún identificador de mensaje, el método devuelve un valor nulo.

Excepciones:

- XMSException

JMSPriority - Obtener y establecer JMSPriority

Interfaz:

```
Int32 JMSPriority
{
  get;
  set;
}
```

Obtener y establecer la prioridad del mensaje.

La prioridad se establece mediante la llamada IMessageProducer.send() cuando se envía el mensaje. El valor es un entero dentro del rango de 0, la prioridad inferior, a 9, la prioridad superior.

Para un mensaje recién creado que no se ha enviado, la prioridad es 4, a menos que la aplicación emisora establezca una prioridad diferente estableciendo JMSPriority. Para un mensaje que se ha recibido, el

método devuelve la prioridad que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie la prioridad estableciendo JMSPriority.

Excepciones:

- XMSException

JMSRedelivered - Obtener y establecer JMSRedelivered

Interfaz:

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

Obtener una indicación de si el mensaje se está volviendo a entregar e indicar si el mensaje se está volviendo a entregar. La indicación se establece mediante la llamada IMessageConsumer.receive() cuando se recibe el mensaje.

Esta propiedad tiene los valores siguientes:

- `True`, si el mensaje se está volviendo a entregar.
- `False`, si el mensaje no se está volviendo a entregar.

Para una conexión en tiempo real a un intermediario, el valor siempre es `False`.

Una indicación de reentrega establecida por JMSRedelivered antes de que se envíe el mensaje es ignorada por la llamada IMessageProducer.send() cuando se envía el mensaje, y la llamada IMessageConsumer.receive() la ignora y la sustituye cuando se recibe el mensaje. Sin embargo, puede utilizar JMSRedelivered para cambiar la indicación para un mensaje que se ha recibido.

Excepciones:

- XMSException

JMSReplyTo - Obtener y establecer JMSReplyTo

Interfaz:

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

Obtener y establecer el destino adonde se va a enviar una respuesta al mensaje.

El valor de esta propiedad es un objeto Destination para el destino adonde se va a enviar una respuesta al mensaje. Un objeto Destination nulo significa que no se espera ninguna respuesta.

Excepciones:

- XMSException

JMSTimestamp - Obtener y establecer JMSTimestamp

Interfaz:

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

Obtener y establecer la hora cuando se envió el mensaje.

La indicación de fecha y hora se establece mediante la llamadaIMessageProducer.send() cuando se envía el mensaje y se expresa en milisegundos desde las 00:00:00 GMT del 1 de enero de 1970.

Para un mensaje recién creado que no se ha enviado, la indicación de fecha y hora es 0, a menos que la aplicación emisora establezca una indicación de fecha y hora diferente estableciendo JMSTimestamp. Para un mensaje que se ha recibido, el método devuelve la indicación de fecha y hora que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie la indicación de fecha y hora estableciendo JMSTimestamp.

Excepciones:

- XMSException

Notas:

1. Si la indicación de fecha y hora no está definida, el método devuelve 0 pero no lanza ninguna excepción.

JMSType - Obtener y establecer JMSType

Interfaz:

```
String JMSType
{
  get;
  set;
}
```

Obtener y establecer el tipo de mensaje.

El valor de JMSType es una serie que encapsula el tipo del mensaje. Si es necesaria una conversión de datos, este valor es el tipo después de la conversión.

Excepciones:

- XMSException

PropertyNames - Obtener propiedades

Interfaz:

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

Obtener una enumeración de las propiedades de nombres del mensaje.

Excepciones:

- XMSException

Métodos

Acknowledge - Acusar recibo

Interfaz:

```
void  Acknowledge();
```

Acusar recibo de este mensaje y todos los mensajes recibidos previamente con acuse de recibo recibidos por la sesión.

Una aplicación puede llamar a este método si la modalidad de acuse de recibo de la sesión es AcknowledgeMode.ClientAcknowledge. Se ignoran las llamadas al método si la sesión tiene cualquier otra modalidad de acuse de recibo o es una sesión con transacción.

Los mensajes que se han recibido pero que no tienen acuse de recibo se podrían volver a entregar.

Si desea más información sobre cómo acusar recibo de mensajes, consulte ../com.ibm.mq.dev.doc/xms_cmesack.dita#xms_cmesack.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- IllegalStateException

ClearBody - Borrar cuerpo

Interfaz:

```
void  ClearBody();
```

Borrar el cuerpo del mensaje. Los campos de cabecera y las propiedades de mensaje no se borran.

Si una aplicación borra un cuerpo del mensaje, el cuerpo permanece en el mismo estado que un cuerpo vacío en un mensaje recién creado. El estado de un cuerpo vacío en un mensaje recién creado depende del tipo de cuerpo del mensaje. Para obtener más información, consulte El cuerpo de un mensaje XMS.

Una aplicación puede borrar un cuerpo del mensaje en cualquier momento, independientemente del estado en el que se encuentra el cuerpo. Si un cuerpo del mensaje es de solo lectura, la única forma en la que una aplicación puede escribir en el cuerpo es que la aplicación borre primero el cuerpo.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

ClearProperties - Borrar propiedades

Interfaz:

```
void  ClearProperties();
```

Borrar las propiedades del mensaje. Los campos de cabecera y el cuerpo del mensaje no se borran.

Si una aplicación borra las propiedades de un mensaje, las propiedades pasan a poderse leer y escribir.

Una aplicación puede borrar las propiedades de un mensaje en cualquier momento, independientemente del estado en que están las propiedades. Si las propiedades de un mensaje son de solo lectura, la única forma en la que las propiedades se pueden grabar es que la aplicación borre primero las propiedades.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

PropertyExists - Comprobar si existe la propiedad

Interfaz:

```
Boolean PropertyExists(String propertyName);
```

Comprobar si el mensaje tiene una prioridad con el nombre especificado.

Parámetros:

propertyName (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

- `True`, si el mensaje tiene una propiedad con el nombre especificado.
- `False`, si el mensaje no tiene una propiedad con el nombre especificado.

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessageConsumer

Una aplicación utiliza un consumidor de mensaje para recibir mensajes enviados a un destino.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

Para obtener una lista de las propiedades definidas de XMS de un objeto MessageConsumer, consulte “Propiedades de MessageConsumer” en la página 2100.

Propiedades de .NET

MessageListener - Obtener y establecer escucha de mensajes

Interfaz:

```
MessageListener MessageListener
{
  get;
  set;
}
```

Obtener el escucha de mensajes que está registrado con el consumidor de mensajes y registrar un escucha de mensajes con el consumidor de mensajes.

Si no hay ningún escucha de mensajes registrado con el consumidor de mensajes, MessageListener es nulo. Si un escucha de mensajes ya está registrado con el consumidor de mensajes, puede cancelar el registro especificando en su lugar un valor nulo.

Para obtener más información sobre cómo utilizar escuchas de mensajes, consulte Utilización de escuchas de mensajes y excepciones en .NET.

Excepciones:

- XMSException

MessageSelector - Obtener selector de mensajes

Interfaz:

```
String MessageSelector
{
  get;
}
```

Obtener el selector de mensajes para el consumidor de mensajes. el valor de retorno es un objeto String que encapsula la expresión de selector de mensajes. Si es necesaria la conversión de datos, este valor es la expresión de selector de mensajes después de la conversión. Si el consumidor de mensajes no tiene un selector de mensajes, el valor de MessageSelector es un objeto String nulo.

Excepciones:

- XMSException

Métodos

Close - Cerrar consumidor de mensajes

Interfaz:

```
void  Close();
```

Cerrar el consumidor de mensajes.

Si una aplicación intenta cerrar un consumidor de mensajes que ya está cerrado, la llamada se ignora.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

Recibir - Recibir

Interfaz:

```
IMessage Receive();
```

Reciba el siguiente mensaje para el consumidor de mensajes. La llamada espera un mensaje de forma indefinida, o hasta que se cierra el consumidor de mensajes.

Parámetros:
Ninguna

Devuelve:
Un puntero del objeto Message. Si el consumidor de mensajes se cierra cuando la llamada está esperando un mensaje, el método devuelve un puntero de un objeto Message nulo.

Excepciones:

- XMSException

Recibir - Recibir (con un intervalo de espera)

Interfaz:

```
IMessage Receive(Int64 delay);
```

Reciba el siguiente mensaje para el consumidor de mensajes. La llamada solo espera un mensaje durante un periodo de tiempo especificado, o hasta que se cierra el consumidor de mensajes.

Parámetros:

delay (entrada)
El tiempo, en milisegundos, que espera un mensaje la llamada. Si especifica un intervalo de espera de 0, la llamada espera un mensaje de forma indefinida.

Devuelve:
Un puntero del objeto Message. Si no llega ningún mensaje durante el intervalo de espera, o si el consumidor de mensajes se cierra mientras la llamada está esperando un mensaje, el método devuelve un puntero de un objeto Message nulo, pero no lanza ninguna excepción.

Excepciones:

- XMSException

ReceiveNoWait - Recibir sin espera

Interfaz:

```
IMessage ReceiveNoWait();
```

Recibir el siguiente mensaje para el consumidor de mensajes si uno está disponible de forma inmediata.

Parámetros:
Ninguna

Devuelve:
Un puntero de un objeto Message. Si no hay ningún mensaje disponible de forma inmediata, el método devuelve un puntero de un objeto Message nulo.

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

MessageEOFException

XMS emite esta excepción si XMS encuentra el final de una corriente de mensajes de bytes cuando una aplicación está leyendo el cuerpo de un mensaje de bytes.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageEOFException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

MessageFormatException

XMS emite esta excepción si XMS encuentra un mensaje con un formato que no es válido.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageFormatException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

IMessageListener (delegado)

Una aplicación utiliza un escucha de mensajes para recibir mensajes de forma asíncrona.

Jerarquía de herencia:
Ninguna

Delegado

MessageListener - Escucha de mensajes

Interfaz:

```
public delegate void MessageListener(IMessage msg);
```

Entregue un mensaje de forma asíncrona al consumidor de mensajes.

Los métodos que implementan este delegado se pueden registrar con la conexión.

Para obtener más información sobre cómo utilizar escuchas de mensajes, consulte Utilización de escuchas de mensajes y excepciones en .NET.

Parámetros:

mesg (entrada)
El objeto Message.

Devuelve:
Void

MessageNotReadableException

XMS emite esta excepción si una aplicación intenta leer el cuerpo de un mensaje que es sólo de escritura.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotReadableException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

MessageNotWritableException

XMS emite esta excepción si una aplicación intenta grabar en el cuerpo de un mensaje que es de sólo lectura.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotWritableException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

IMessageProducer

Una aplicación utiliza un productor de mensajes para enviar mensajes a un destino.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

Si desea una lista de las propiedades definidas de XMS de un objeto MessageProducer, consulte “Propiedades de MessageProducer” en la página 2100.

Propiedades de .NET

DeliveryMode - Obtener y establecer modalidad de entrega predeterminada

Interfaz:

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

Obtener y establecer la modalidad de entrega predeterminada para mensajes enviados por el productor de mensajes.

La modalidad de entrega predeterminada tiene uno de los valores siguientes:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Para una conexión en tiempo real con un intermediario, el valor debe ser `DeliveryMode.NonPersistent`.

El valor predeterminado es `DeliveryMode.Persistent`, excepto para una conexión en tiempo real con un intermediario para el cual el valor predeterminado es `DeliveryMode.NonPersistent`.

Excepciones:

- XMSException

Destination - Obtener destino

Interfaz:

```
IDestination Destination
{
  get;
}
```

Obtener el destino para el productor de mensajes.

Parámetros:
Ninguna

Devuelve:
El objeto Destination. Si el productor de mensajes no tiene un destino, el método devuelve un objeto Destination nulo.

Excepciones:

- XMSException

DisableMsgID - Obtener y establecer Inhabilitar distintivo de ID de mensaje

Interfaz:

```
Boolean DisableMessageID
{
  get;
  set;
}
```

Obtener una indicación sobre si una aplicación receptora requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes, e indicar si una aplicación receptora requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes.

En una conexión con un gestor de colas, o en una conexión en tiempo real con un intermediario, este distintivo se ignora. En una conexión con un bus de integración de servicios, se respeta el distintivo.

DisabledMsgID tiene los valores siguientes:

- `True`, si una aplicación receptora no requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes.
- `False`, si una aplicación requiere que se incluyan identificadores de mensajes en los mensajes enviados por el productor de mensajes.

Excepciones:

- XMSException

DisableMsgTS - Obtener y establecer Inhabilitar distintivo de indicación de fecha y hora

Interfaz:

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

Obtener una indicación sobre si una aplicación receptora requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes, e indicar si una aplicación receptora

requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.

En una conexión de tiempo real con un intermediario, se ignora este distintivo. En una conexión con un gestor de colas, o en una conexión con un bus de integración de servicios, se respeta el distintivo.

DisableMsgTS tiene los valores siguientes:

- `True`, si una aplicación receptora no requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.
- `False`, si una aplicación receptora requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.

Devuelve:

Excepciones:

- XMSException

Priority - Obtener y establecer prioridad predeterminada

Interfaz:

```
Int32 Priority
{
  get;
  set;
}
```

Obtener y establecer la prioridad predeterminada para los mensajes enviados por el productor de mensajes.

El valor de la prioridad de mensaje predeterminada es un entero dentro del rango de `0`, la prioridad más baja, a 9, la prioridad más alta.

En una conexión en tiempo real con un intermediario, se ignora la prioridad de un mensaje.

Excepciones:

- XMSException

TimeToLive - Obtener y establecer tiempo de vida

Interfaz:

```
Int64 TimeToLive
{
  get;
  set;
}
```

Obtener y establecer el periodo de tiempo predeterminado que existe un mensaje antes de caducar.

El tiempo se mide a partir del momento cuando el productor de mensajes envía el mensaje y el tiempo de vida predeterminado en milisegundos. Un valor de 0 indica que un mensaje no caduca nunca.

Para una conexión en tiempo real con un intermediario, este valor siempre es 0.

Excepciones:

- XMSException

Métodos

Close - Cerrar productor de mensajes

Interfaz:

```
void  Close();
```

Cerrar el productor de mensajes.

Si una aplicación intenta cerrar un productor de mensajes que ya está cerrado, se ignora la llamada.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

Send - Enviar

Interfaz:

```
void Send(IMessage msg) ;
```

Enviar un mensaje al destino que se ha especificado al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, la prioridad y el tiempo de vida predeterminados del productor de mensajes.

Parámetros:

msg (entrada)
El objeto Message.

Devuelve:
Void

Excepciones:

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - Enviar (especificando una modalidad de entrega, prioridad y tiempo de vida)

Interfaz:

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

Enviar un mensaje al destino que se ha especificado al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, prioridad y tiempo de vida especificados.

Parámetros:

msg (entrada)
El objeto Message.

deliveryMode (entrada)
La modalidad de entrega para el mensaje, que debe adoptar uno de los valores siguientes:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Para una conexión en tiempo real con un intermediario, el valor debe ser `DeliveryMode.NonPersistent`.

priority (entrada)
La prioridad del mensaje. El valor puede ser un entero dentro del rango de `0`, para la prioridad más baja, a 9, para la prioridad más alta. En una conexión en tiempo real con un intermediario, se ignora el valor.

timeToLive (entrada)
El tiempo de vida para el mensaje en milisegundos. Un valor de 0 significa que el mensaje no caduca nunca. Para una conexión en tiempo real con un intermediario, el valor debe ser 0.

Devuelve:
Void

Excepciones:

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Send - Enviar (a un destino especificado)

Interfaz:

```
void Send(IDestination dest, IMessage msg) ;
```

Enviar un mensaje a un destino especificado si está utilizando un productor de mensajes para el que no se ha especificado ningún destino al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, la prioridad y el tiempo de vida predeterminados del productor de mensajes.

Normalmente, puede especificar un destino cuando crea un productor de mensajes, pero si no lo hace, debe especificar un destino cada vez que envíe un mensaje.

Parámetros:

dest (entrada)
El objeto Destination.

msg (entrada)
El objeto Message.

Devuelve:
Void

Excepciones:

- XMSException
- MessageFormatException
- InvalidDestinationException

Send - Enviar (a un destino especificado, especificando una modalidad de entrega, prioridad y tiempo de vida)

Interfaz:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Enviar un mensaje a un destino especificado si está utilizando un productor de mensajes para el que no se ha especificado ningún destino al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, prioridad y tiempo de vida especificados.

Normalmente, puede especificar un destino cuando crea un productor de mensajes, pero si no lo hace, debe especificar un destino cada vez que envíe un mensaje.

Parámetros:

dest (entrada)
El objeto Destination.

msg (entrada)
El objeto Message.

deliveryMode (entrada)
La modalidad de entrega para el mensaje, que debe adoptar uno de los valores siguientes:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Para una conexión en tiempo real con un intermediario, el valor debe ser `DeliveryMode.NonPersistent`.

priority (entrada)
La prioridad del mensaje. El valor puede ser un entero dentro del rango de `0`, para la prioridad más baja, a 9, para la prioridad más alta. En una conexión en tiempo real con un intermediario, se ignora el valor.

timeToLive (entrada)
El tiempo de vida para el mensaje en milisegundos. Un valor de 0 significa que el mensaje no caduca nunca. Para una conexión en tiempo real con un intermediario, el valor debe ser 0.

Devuelve:
Void

Excepciones:

- XMSException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IObjectMessage

Un mensaje de objeto es un mensaje cuyo cuerpo está formado por un objeto Java o .NET serializado.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
            |
            +----IBM.XMS.IObjectMessage
```

Propiedades de .NET

Object - Obtener y establecer objeto como bytes

Interfaz:

```
System.Object Object
    {
      get;
      set;
    }

    Byte[] GetObject();
```

Obtener y establecer el objeto que forma el cuerpo del mensaje de objeto.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody, clearProperties, PropertyExists

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

IPropertyContext es una superclase abstracta que contiene métodos que obtienen y establecen propiedades. Estos métodos son heredados por otras clases.

Jerarquía de herencia:
Ninguna

Métodos

GetBooleanProperty - Obtener propiedad booleana

Interfaz:

```
Boolean GetBooleanProperty(String property_name);
```

Obtener el valor de la propiedad booleana con el nombre especificado.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetByteProperty - Obtener propiedad de byte

Interfaz:

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

Obtener el valor de la propiedad de byte identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetBytesProperty - Obtener propiedad de matriz de bytes

Interfaz:

```
Byte[]  GetBytesProperty(String property_name) ;
```

Obtener el valor de la propiedad de matriz de bytes identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El número de bytes de la matriz.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetCharProperty - Obtener propiedad de carácter

Interfaz:

```
Char    GetCharProperty(String property_name) ;
```

Obtener el valor de la propiedad de carácter de 2-bytes identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetDoubleProperty - Obtener propiedad de coma flotante de precisión doble

Interfaz:

```
Double  GetDoubleProperty(String property_name) ;
```

Obtener el valor de la propiedad de coma flotante de precisión doble identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetFloatProperty - Obtener propiedad de coma flotante

Interfaz:

```
Single  GetFloatProperty(String property_name) ;
```

Obtener el valor de la propiedad de coma flotante identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetIntProperty - GetIntProperty

Interfaz:

```
Int32   GetIntProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetLongProperty - Obtener propiedad de entero largo

Interfaz:

```
Int64   GetLongProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero largo identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetObjectProperty - Obtener propiedad de objeto

Interfaz:

```
Object  GetObjectProperty( String property_name) ;
```

Obtener el valor y el tipo de datos de la propiedad identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad, que adopta uno de los tipos de objeto siguientes:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetShortProperty - Obtener propiedad de entero corto

Interfaz:

```
Int16   GetShortProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero corto identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
El valor de la propiedad.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

GetStringProperty - GetStringProperty

Interfaz:

```
String  GetStringProperty(String property_name) ;
```

Obtener el valor de la propiedad de serie identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

Devuelve:
Un objeto String que encapsula la serie que es el valor de la propiedad. Si es necesaria una conversión de datos, este valor es la serie después de la conversión.

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException

SetBooleanProperty - Establecer propiedad booleana

Interfaz:

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

Establecer el valor de la propiedad booleana identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetByteProperty - Establecer propiedad de byte

Interfaz:

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

Establecer el valor de la propiedad de byte identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetBytesProperty - Establecer propiedad de matriz de bytes

Interfaz:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Establecer el valor de la propiedad de matriz de bytes identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad, que es una matriz de bytes.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetCharProperty - Establecer propiedad de carácter

Interfaz:

```
void SetCharProperty( String property_name,  Char value) ;
```

Establecer el valor de la propiedad de carácter de 2-bytes identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetDoubleProperty - Establecer propiedad de coma flotante de precisión doble

Interfaz:

```
void SetDoubleProperty( String property_name,  Double value) ;
```

Establecer el valor de la propiedad de coma flotante de precisión doble identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetFloatProperty - Establecer propiedad de coma flotante

Interfaz:

```
void SetFloatProperty( String property_name,  Single value) ;
```

Establecer el valor de la propiedad de coma flotante identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetIntProperty - Establecer propiedad de entero

Interfaz:

```
void SetIntProperty( String property_name,  Int32 value) ;
```

Establecer el valor de la propiedad de entero identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetLongProperty - Establecer propiedad de entero largo

Interfaz:

```
void SetLongProperty( String property_name,  Int64 value) ;
```

Establecer el valor de la propiedad de entero largo identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetObjectProperty - Establecer propiedad de objeto

Interfaz:

```
void SetObjectProperty( String property_name,  Object value) ;
```

Establecer el valor y el tipo de datos de una propiedad identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

objectType (entrada)
El valor de la propiedad, que debe adoptar uno de los tipos de objeto siguientes:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (entrada)
El valor de la propiedad como una matriz de bytes.

length (entrada)
El número de bytes de la matriz.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetShortProperty - Establecer propiedad de entero corto

Interfaz:

```
void SetShortProperty( String property_name,  Int16 value) ;
```

Establecer el valor de la propiedad de entero corto identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
El valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

SetStringProperty - Establecer propiedad de serie

Interfaz:

```
void SetStringProperty( String property_name,  String value);
```

Establecer el valor de la propiedad de serie identificada por el nombre.

Parámetros:

property_name (entrada)
Un objeto String que encapsula el nombre de la propiedad.

value (entrada)
Un objeto String que encapsula la serie que es el valor de la propiedad.

Devuelve:
Void

Contexto de hebras
Se determina mediante la subclase.

Excepciones:

- XMSException
- MessageNotWritableException

IQueueBrowser

Una aplicación utiliza un examinador de colas para examinar mensajes en una cola sin eliminarlas.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
   |
   +----IBM.XMS.IQueueBrowser
```

Propiedades de .NET

MessageSelector - Obtener selector de mensajes

Interfaz:

```
String MessageSelector
{
  get;
}
```

Obtener el selector de mensajes para el examinador de colas.

El selector de mensajes es un objeto String que encapsula la expresión de selector de mensajes. Si es necesaria la conversión de datos, este valor es la expresión de selector de mensajes después de la conversión. Si el examinador de colas no tiene un selector de mensajes, el método devuelve un objeto String nulo.

Excepciones:

- XMSException

Queue - Obtener cola

Interfaz:

```
IDestination Queue
```

```
{
  get;
}
```

Obtener la cola asociada al examinador de colas como un objeto de destino que representa la cola.

Excepciones:

- XMSException

Métodos

Close - Cerrar examinador de colas

Interfaz:

```
void  Close();
```

Cerrar el examinador de colas.

Si una aplicación intenta cerrar un examinador de colas que está cerrado, la llamada se ignora.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException

GetEnumerator - Obtener mensajes

Interfaz:

```
IEnumerator GetEnumerator();
```

Obtener una lista de los mensajes de la cola.

El método devuelve un enumerador que encapsula una lista de objetos Message. El orden de los objetos Message es el mismo que el orden en el cual se recuperarán los mensajes de la cola. La aplicación puede utilizar el enumerador para examinar cada mensaje a su vez.

El enumerador se actualiza de forma dinámica a medida que los mensajes se colocan en la cola y se eliminan de la cola. Cada vez que la aplicación llama a IEnumerator.MoveNext() para examinar el siguiente mensaje en la cola, el mensaje refleja el contenido actual de la cola.

Si una aplicación llama a este método más de una vez para un examinador de colas, cada llamada devuelve un nuevo enumerador. Por lo tanto, la aplicación puede utilizar más de un enumerador para examinar los mensajes de una cola y mantener varias posiciones dentro de la cola.

Parámetros:
Ninguna

Devuelve:
El objeto Iterator.

Excepciones:

- XMSException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Solicitante

Una aplicación utiliza un solicitante para enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla.

Jerarquía de herencia:
Ninguna

Constructores

Requestor - Crear solicitante

Interfaz:

```
Requestor(ISession sess, IDestination dest);
```

Crear un solicitante.

Parámetros:

sess (entrada)
Un objeto Session. La sesión no debe ser una sesión con transacción y debe tener una de las modalidades de acuse de recibo siguientes:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

dest (entrada)
Un objeto Destination que representa el destino adonde la aplicación puede enviar mensajes de solicitud.

Contexto de hebras
La sesión asociada al solicitante

Excepciones:

- XMSException

Métodos

Close - Cerrar solicitante

Interfaz:

```
void  Close();
```

Cerrar el solicitante.

Si una aplicación intenta cerrar un solicitante que ya está cerrado, se ignora la llamada.

Nota: Cuando una aplicación cierra un solicitante, la sesión asociada no se cierra. En este sentido, XMS se comporta de forma diferente en comparación con JMS.

Parámetros:
Ninguna

Devuelve:
Void

Contexto de hebras
Cualquiera

Excepciones:

- XMSException

Solicitar - Solicitar respuesta

Interfaz:

```
IMessage Request(IMessage requestMessage);
```

Enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla de la aplicación que recibe el mensaje de solicitud.

Se bloquea una llamada a este método hasta que se recibe una respuesta o hasta que finaliza la sesión, lo que se produzca antes.

Parámetros:

requestMessage (entrada)
El objeto Message que encapsula el mensaje de solicitud.

Devuelve:
Un puntero del objeto Message que encapsula el mensaje de respuesta.

Contexto de hebras
La sesión asociada al solicitante

Excepciones:

- XMSException

ResourceAllocationException

XMS emite esta excepción si XMS no puede asignar los recursos necesarios para un método.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

SecurityException

XMS emite esta excepción si se rechazan el identificador de usuario y la contraseña proporcionados para autenticar una aplicación. XMS también lanza esta excepción si una comprobación de autoridad falla e impide que se complete un método.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

ISession

Una sesión es un contexto de hebra única para enviar y recibir mensajes.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

Si desea una lista de propiedades definidas de XMS de un objeto Session, consulte "Propiedades de sesión" en la página 2100.

Propiedades de .NET

AcknowledgeMode - Obtener modalidad de acuse de recibo

Interfaz:

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

Obtener la modalidad de acuse de recibo para la sesión.

La modalidad de acuse de recibo se especifica cuando se crea la sesión.

Si la sesión no es una sesión con transacción, la modalidad de acuse de recibo es uno de los valores siguientes:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Para obtener más información sobre las modalidades de acuse de recibo, consulte Acuse de recibo de mensaje.

Una sesión que es una sesión con transacción no tiene ninguna modalidad de acuse de recibo. Si la sesión es una sesión con transacción, en su lugar el método devuelve `AcknowledgeMode.SessionTransacted`.

Excepciones:

- XMSException

Transacted - Determinar si es transaccional

Interfaz:

```
Boolean Transacted
    {
      get;
    }
```

Determinar si la sesión es una sesión con transacción.

El estado con transacción es:

- Verdadero, si la sesión es transaccional.

- Falso, si la sesión no es transaccional.

Para una conexión en tiempo real con un intermediario, el método siempre devuelve False.

Excepciones:

- XMSException

Métodos

Close - Cerrar sesión

Interfaz:

```
void Close();
```

Cerrar la sesión. Si la sesión es una sesión con transacción, se retrotrae cualquier transacción en curso.

Si una aplicación intenta cerrar una sesión que ya está cerrada, la llamada se ignora.

Parámetros:
Ninguna

Devuelve:
Void

Contexto de hebras
Cualquiera

Excepciones:

- XMSException

Commit - Confirmar

Interfaz:

```
void Commit();
```

Confirmar todos los mensajes procesados en la transacción actual.

La sesión debe ser una sesión con transacción.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - Crear examinador de colas

Interfaz:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Crear un examinador de colas para la cola especificada.

Parámetros:

queue (entrada)
Un objeto Destination que representa la cola.

Devuelve:
El objeto QueueBrowser.

Excepciones:

- XMSException
- InvalidDestinationException

CreateBrowser - Crear examinador de colas (con selector de mensajes)

Interfaz:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Crear un examinador de colas para la cola especificada utilizando un selector de mensajes.

Parámetros:

queue (entrada)
Un objeto Destination que representa la cola.

selector (entrada)
Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al examinador de colas los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el examinador de colas.

Devuelve:
El objeto QueueBrowser.

Excepciones:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - Crear mensaje de bytes

Interfaz:

```
IBytesMessage CreateBytesMessage();
```

Crear un mensaje de bytes.

Parámetros:
Ninguna

Devuelve:
El objeto BytesMessage.

Excepciones:

- XMSException
- IllegalStateException (La sesión está cerrada)

CreateConsumer - Crear consumidor

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Crear un consumidor de mensajes para el destino especificado.

Parámetros:

dest (entrada)
El objeto Destination.

Devuelve:
El objeto MessageConsumer.

Excepciones:

- XMSException
- InvalidDestinationException

CreateConsumer - Crear consumidor (con selector de mensajes)

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

Crear un consumidor de mensajes para el destino especificado utilizando un selector de mensajes.

Parámetros:

dest (entrada)
El objeto Destination.

selector (entrada)
Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al consumidor de mensajes los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el consumidor de mensajes.

Devuelve:
El objeto MessageConsumer.

Excepciones:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - Crear consumidor (con el selector de mensajes y el distintivo de mensaje local)

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

Crear un consumidor de mensajes para el destino especificado utilizando un selector de mensajes y, si el destino es un tema, especificando si el consumidor de mensajes recibe los mensajes publicados por su propia conexión.

Parámetros:

dest (entrada)
El objeto Destination.

selector (entrada)
Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al consumidor de mensajes los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el consumidor de mensajes.

noLocal (entrada)
El valor True significa que el consumidor de mensajes no recibe los mensajes publicados por su propia conexión. El valor False significa que el consumidor de mensajes no recibe los mensajes publicados por su propia conexión. El valor predeterminado es False.

Devuelve:
El objeto MessageConsumer.

Excepciones:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateDurableSubscriber - Crear suscriptor duradero

Interfaz:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

Crear un suscriptor duradero para el tema especificado.

Este método no es válido para una conexión en tiempo real con un intermediario.

Para obtener más información sobre los suscriptores duraderos, consulte Suscriptores duraderos.

Parámetros:

dest (entrada)
Un objeto Destination que representa el tema. El tema no debe ser un tema temporal.

subscription (entrada)
Un objeto String encapsula un nombre que identifica la suscripción duradera. El nombre debe ser exclusivo dentro del identificador de cliente para la conexión.

Devuelve:
El objeto MessageConsumer que representa el suscriptor duradero.

Excepciones:

- XMSException
- InvalidDestinationException

CreateDurableSubscriber - Crear suscriptores duraderos (con selector de mensajes y distintivo de mensaje local)

Interfaz:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

Crear un suscriptor duradero para el tema especificado utilizando un selector de mensajes y especificando si el suscriptor duradero recibe los mensajes publicados por su propia conexión.

Este método no es válido para una conexión en tiempo real con un intermediario.

Para obtener más información sobre los suscriptores duraderos, consulte Suscriptores duraderos.

Parámetros:

dest (entrada)
Un objeto Destination que representa el tema. El tema no debe ser un tema temporal.

subscription (entrada)
Un objeto String encapsula un nombre que identifica la suscripción duradera. El nombre debe ser exclusivo dentro del identificador de cliente para la conexión.

selector (entrada)
Un objeto String que encapsula una expresión de selector de mensajes. Solo se devuelven al suscriptor duradero los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el suscriptor duradero.

noLocal (entrada)
El valor True significa que el suscriptor duradero no recibe los mensajes publicados por su propia conexión. El valor False significa que el suscriptor duradero no recibe los mensajes publicados por su propia conexión. El valor predeterminado es False.

Devuelve:
El objeto MessageConsumer que representa el suscriptor duradero.

Excepciones:

- XMSException
- InvalidDestinationException
- InvalidSelectorException

CreateMapMessage - Crear mensaje de correlación

Interfaz:

```
IMapMessage CreateMapMessage();
```

Crear un mensaje de correlación.

Parámetros:
Ninguna

Devuelve:
El objeto MapMessage.

Excepciones:

- XMSException
- IllegalStateException (La sesión está cerrada)

CreateMessage - Crear mensaje

Interfaz:

```
IMessage CreateMessage();
```

Crear un mensaje que no tiene un cuerpo.

Parámetros:
Ninguna

Devuelve:
El objeto Message.

Excepciones:

- XMSException
- IllegalStateException (La sesión está cerrada)

CreateObjectMessage - Crear mensaje de objeto

Interfaz:

```
IObjectMessage CreateObjectMessage();
```

Crear un mensaje de objeto.

Parámetros:
Ninguna

Devuelve:
El objeto ObjectMessage.

Excepciones:

- XMSException
- IllegalStateException (La sesión está cerrada)

CreateProducer - Crear productor

Interfaz:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Crear un productor de mensajes para enviar mensajes al destino especificado.

Parámetros:

dest (entrada)
El objeto Destination.

Si especifica un objeto Destination nulo, el productor de mensajes se crea sin un destino. En este caso, la aplicación debe especificar un destino cada vez que utiliza el productor de mensajes envía un mensaje.

Devuelve:
El objeto MessageProducer.

Excepciones:

- XMSException
- InvalidDestinationException

CreateQueue - Crear cola

Interfaz:

```
IDestination CreateQueue(String queue) ;
```

Crear un objeto Destination para representar una cola en el servidor de mensajería.

Este método no crea la cola en el servidor de mensajería. Debe crear la cola antes de que una aplicación pueda llamar a este método.

Parámetros:

queue (entrada)
Un objeto String que encapsula el nombre de la cola, o encapsula un identificador uniforme de recursos (URI) que identifica la cola.

Devuelve:
El objeto Destination que representa la cola.

Excepciones:

- XMSException

CreateStreamMessage - Crear mensaje de corriente de datos

Interfaz:

```
IStreamMessage CreateStreamMessage();
```

Crear un mensaje de corriente de datos.

Parámetros:
Ninguna

Devuelve:
El objeto StreamMessage.

Excepciones:

- XMSException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - Crear cola temporal

Interfaz:

```
IDestination CreateTemporaryQueue() ;
```

Crear una cola temporal.

El ámbito de la cola temporal es la conexión. Solo las sesiones creadas por la conexión pueden utilizar la cola temporal.

La cola temporal permanece hasta que se suprime de forma explícita, o finaliza la conexión, lo que suceda antes.

Para obtener más información sobre las colas temporales, consulte Destinos temporales.

Parámetros:
Ninguna

Devuelve:
El objeto Destination que representa la cola temporal.

Excepciones:

- XMSException

CreateTemporaryTopic - Crear tema temporal

Interfaz:

```
IDestination CreateTemporaryTopic() ;
```

Crear un tema temporal.

El ámbito del tema temporal es la conexión. Solo las sesiones creadas por la conexión pueden utilizar el tema temporal.

El tema temporal permanece hasta que se suprime de forma explícita, o finaliza la conexión, lo que suceda antes.

Para obtener más información sobre temas temporales, consulte Destinos temporales.

Parámetros:
Ninguna

Devuelve:
El objeto Destination que representa el tema temporal.

Excepciones:

- XMSException

CreateTextMessage - Crear mensaje de texto

Interfaz:

```
ITextMessage CreateTextMessage();
```

Crear un mensaje de texto con un cuerpo vacío.

Parámetros:
Ninguna

Devuelve:
El objeto TextMessage.

Excepciones:

- XMSException

CreateTextMessage - Crear mensaje de texto (inicializado)

Interfaz:

```
ITextMessage CreateTextMessage(String initialValue);
```

Crear un mensaje de texto cuyo cuerpo se ha inicializado con el texto especificado.

Parámetros:

initialValue (entrada)
Un objeto String que encapsula el texto para inicializar el cuerpo del mensaje de texto.

Ninguna

Devuelve:
El objeto TextMessage.

Excepciones:

- XMSException

CreateTopic - Crear tema

Interfaz:

```
IDestination CreateTopic(String topic) ;
```

Crear un objeto Destination para representar un tema.

Parámetros:

topic (entrada)
Un objeto String que encapsula el nombre del tema, o encapsula un identificador uniforme de recursos (URI) que identifica el tema.

Devuelve:
El objeto Destination que representa el tema.

Excepciones:

- XMSException

Recover - Recuperar

Interfaz:

```
void Recover();
```

Recuperar la sesión. Se detiene la entrega de mensaje y, después, se reinicia con el mensaje sin acuse de recibo más antiguo.

La sesión no debe ser una sesión con transacción.

Para obtener más información sobre la recuperación de una sesión, consulte Acuse de recibo de mensaje.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- IllegalStateException

Rollback - Retrotraer

Interfaz:

```
void Rollback();
```

Retrotraer todos los mensajes procesados en la transacción actual.

La sesión debe ser una sesión con transacción.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- IllegalStateException

Unsubscribe - Anular suscripción

Interfaz:

```
void Unsubscribe(String subscription);
```

Suprimir una suscripción duradera. El servidor de mensajería suprime el registro de la suscripción duradera que está manteniendo y no envía ningún mensaje más al suscriptor duradero.

Una aplicación no puede suprimir una suscripción duradera bajo ninguna de las circunstancias siguientes:

- Mientras haya un consumidor de mensajes activo para la suscripción duradera
- Mientras un mensaje consumido forme parte de una transacción pendiente
- Mientras un mensaje consumido no tenga acuse de recibo

Este método no es válido para una conexión en tiempo real con un intermediario.

Parámetros:

subscription (entrada)
Un objeto String que encapsula el nombre que identifica la suscripción duradera.

Devuelve:
Void

Excepciones:

- XMSException
- InvalidDestinationException
- IllegalStateException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IStreamMessage

Un mensaje de corriente de datos es un mensaje cuyo cuerpo está formado por una corriente de valores, donde cada valor tiene un tipo de datos asociado. El contenido del cuerpo se escribe y se lee de forma secuencial.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IStreamMessage
```

Cuando una aplicación lee un valor de la corriente de datos de mensaje, el valor se puede convertir mediante XMS a otro tipo de datos. Para obtener más información sobre esta forma de conversión implícita, consulte El cuerpo de un mensaje de XMS.

Métodos

ReadBoolean - Leer valor booleano

Interfaz:

```
Boolean ReadBoolean();
```

Leer un valor booleano en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El valor booleano que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Leer byte

Interfaz:

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

Leer un entero de 8-bits firmado en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El byte que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Leer bytes

Interfaz:

```
Int32   ReadBytes(Byte[] array);
```

Leer una matriz de bytes en la corriente de datos de mensaje.

Parámetros:

array (entrada)
El almacenamiento intermedio que contiene la matriz de bytes que se ha leído y la longitud del almacenamiento intermedio en bytes.

Si el número de bytes de la matriz es menor o igual que la longitud del almacenamiento intermedio, se lee toda la matriz en el almacenamiento intermedio. Si el número de bytes de la matriz es mayor que la longitud del almacenamiento intermedio, el almacenamiento intermedio se rellena con parte de la matriz, y un cursor interno marca la posición del siguiente byte que se va a leer. Una llamada posterior a readBytes() lee bytes de la matriz empezando por la posición actual del cursor.

Si especifica un puntero nulo en la entrada, la llamada se salta la matriz de bytes sin leerla.

Devuelve:
El número de bytes que se van a leer en el almacenamiento intermedio. Si el almacenamiento intermedio se rellena parcialmente, el valor es menor que la longitud del almacenamiento intermedio, lo que indica que no quedan más bytes en la matriz para leer. Si no queda ningún byte para leer en la matriz antes de la llamada, el valor es `XMSC_END_OF_BYTEARRAY`.

Si especifica un puntero nulo en la entrada, el método no devuelve ningún valor.

Excepciones:

- XMSException

- MessageNotReadableException
- MessageEOFException

ReadChar - Leer carácter

Interfaz:

```
Char    ReadChar();
```

Leer un carácter de 2-bytes en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El carácter que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Leer número de coma flotante de precisión doble

Interfaz:

```
Double  ReadDouble();
```

Leer un número de coma flotante de precisión doble de 8-bytes en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El número de coma flotante de precisión doble que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leer número de coma flotante

Interfaz:

```
Single  ReadFloat();
```

Leer un número de coma flotante de 4-bytes en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El número de coma flotante que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Leer entero

Interfaz:

```
Int32   ReadInt();
```

Leer un entero de 32-bits firmado en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El entero que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Leer entero largo

Interfaz:

```
Int64   ReadLong();
```

Leer un entero de 64-bits firmado en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El entero largo que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject - Leer objeto

Interfaz:

```
Object  ReadObject();
```

Leer un valor en la corriente de datos de mensaje y devolver su tipo de datos.

Parámetros:
Ninguna

Devuelve:
El valor, que es uno de los tipos de objeto siguientes:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
```

Int16
String

Excepciones:
XMSException

ReadShort - Leer entero corto

Interfaz:

```
Int16   ReadShort();
```

Leer un entero de 16-bits firmado en la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
El entero corto que se ha leído.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

ReadString - Leer serie

Interfaz:

```
String  ReadString();
```

Leer una serie en la corriente de datos de mensaje. Si es necesario, XMS convierte los caracteres de la serie en la página de códigos local.

Parámetros:
Ninguna

Devuelve:
Un objeto String que encapsula la serie que se ha leído. Si es necesaria la conversión de datos, esta es la serie después de la conversión.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

Reset - Restablecer

Interfaz:

```
void  Reset();
```

Colocar el cuerpo del mensaje en la modalidad de solo lectura y volver a colocar el cursor al principio de la corriente de datos de mensaje.

Parámetros:
Ninguna

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - Escribir valor booleano

Interfaz:

```
void  WriteBoolean(Boolean value);
```

Escribir un valor booleano en la corriente de datos de mensaje.

Parámetros:

value (entrada)
El valor booleano que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteByte - Escribir byte

Interfaz:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Escribir un byte en la corriente de datos de mensaje.

Parámetros:

value (entrada)
El byte que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteBytes - Escribir bytes

Interfaz:

```
void  WriteBytes(Byte[] value);
```

Escribir una matriz de bytes en la corriente de datos de mensaje.

Parámetros:

value (entrada)
La matriz de bytes que se va a escribir.

length (entrada)
El número de bytes de la matriz.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteChar - Escribir carácter

Interfaz:

```
void  WriteChar(Char value);
```

Escribir un carácter en la corriente de datos de mensaje como 2 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El carácter que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteDouble - Escribir número de coma flotante de precisión doble

Interfaz:

```
void  WriteDouble(Double value);
```

Convertir un número de coma flotante de precisión doble a un entero largo y escribir el entero largo en la corriente de datos de mensaje como 8 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El número de coma flotante de precisión doble que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteFloat - Escribir número de coma flotante

Interfaz:

```
void  WriteFloat(Single value);
```

Convertir una número de coma flotante a un entero y escribir el entero en la corriente de datos de mensaje como 4 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El número de coma flotante que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteInt - Escribir entero

Interfaz:

```
void  WriteInt(Int32 value);
```

Escribir un entero en la corriente de datos de mensaje como 4 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteLong - Escribir entero largo

Interfaz:

```
void  WriteLong(Int64 value);
```

Escribir un entero largo en la corriente de datos de mensaje como 8 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero largo que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteObject - Escribir objeto

Interfaz:

```
void  WriteObject(Object value);
```

Escribir un valor, con un tipo de datos especificado, en la corriente de datos de mensaje.

Parámetros:

objectType (entrada)
El valor, que debe adoptar uno de los tipos de objeto siguientes:

`Boolean`
`Byte`

```
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

value (entrada)
Una matriz de bytes que contiene el valor que se va a escribir.

length (entrada)
El número de bytes de la matriz.

Devuelve:
Void

Excepciones:

- XMSException

WriteShort - Escribir entero corto

Interfaz:

```
void  WriteShort(Int16 value);
```

Escribir un entero corto en la corriente de datos de mensaje como 2 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)
El entero corto que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

WriteString - Escribir serie

Interfaz:

```
void  WriteString(String value);
```

Escribir serie en la corriente de datos de mensaje.

Parámetros:

value (entrada)
Un objeto String que encapsula la serie que se va a escribir.

Devuelve:
Void

Excepciones:

- XMSException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody, clearProperties, PropertyExists

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ITextMessage

Un mensaje de texto es un mensaje cuyo cuerpo está formado por una serie.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

Propiedades de .NET

Text - Obtener y establecer texto

Interfaz:

```
String Text
{
  get;
  set;
}
```

Obtener y establecer la serie que forma el cuerpo del mensaje de texto.

Si es necesario, XMS convierte los caracteres de la serie en la página de códigos local.

Excepciones:

- XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody, clearProperties, PropertyExists

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

TransactionInProgressException

XMS emite esta excepción si una aplicación solicita una operación que no es válida porque hay una transacción en curso.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

TransactionRolledBackException

XMS emite esta excepción si una aplicación llama a Session.commit() para confirmar la transacción actual, pero la transacción se retrotrae.

Jerarquía de herencia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz XMSException:

GetErrorCode, GetLinkedException

XMSException

Si XMS detecta un error al procesar una llamada a un método .NET , XMS emite una excepción. Una excepción es un objeto que encapsula información sobre el error.

Jerarquía de herencia:

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

Hay distintos tipos de excepción XMS y un objeto XMSException es sólo un tipo de excepción. Sin embargo, la clase XMSException es una superclase de las otras clases de excepción XMS. XMS lanza un objeto XMSException en situaciones donde ninguno de los otros tipos de excepción es apropiado.

Propiedades de .NET

ErrorCode - Obtener código de error

Interfaz:

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

Obtener el código de error.

Excepciones:

- XMSException

LinkedException - Obtener excepción enlazada

Interfaz:

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

Obtener la siguiente excepción de la cadena de excepciones.

El método devuelve un nulo, si no hay más excepciones en la cadena.

Excepciones:

- XMSException

XMSFactoryFactory

Si una aplicación no está utilizando objetos administrados, utilice esta clase para crear fábricas de conexiones, colas y temas.

Jerarquía de herencia:
Ninguna

Propiedades de .NET

Metadata - Recuperar metadatos

Interfaz:

```
IConnectionMetaData MetaData
```

Obtener los metadatos que son apropiados para el tipo de conexión del objeto XMSFactoryFactory.

Excepciones:
Ninguna

Métodos

CreateConnectionFactory - Crear fábrica de conexiones

Interfaz:

```
IConnectionFactory CreateConnectionFactory();
```

Crear un objeto ConnectionFactory del tipo declarado.

Parámetros:
Ninguna

Devuelve:
El objeto ConnectionFactory.

Excepciones:

- XMSException

CreateQueue - Crear cola

Interfaz:

```
IDestination CreateQueue(String name);
```

Crear un objeto Destination para representar una cola en el servidor de mensajería.

Este método no crea la cola en el servidor de mensajería. Debe crear la cola antes de que una aplicación pueda llamar a este método.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre de la cola, o encapsula un identificador uniforme de recursos (URI) que identifica la cola.

Devuelve:
El objeto Destination que representa la cola.

Excepciones:

- XMSException

CreateTopic - Crear tema

Interfaz:

```
IDestination CreateTopic(String name);
```

Crear un objeto Destination para representar un tema.

Parámetros:

name (entrada)
Un objeto String que encapsula el nombre del tema, o encapsula un identificador uniforme de recursos (URI) que identifica el tema.

Devuelve:
El objeto Destination que representa el tema.

Excepciones:

- XMSException

GetInstance - Obtener una instancia de XMSFactoryFactory

Interfaz:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Cree una instancia de XMSFactoryFactory. Una aplicación XMS utiliza un objeto XMSFactoryFactory para obtener una referencia a un objeto ConnectionFactory que es apropiado para el tipo de protocolo necesario. Este objeto ConnectionFactory puede generar conexiones solo para ese tipo de protocolo.

Parámetros:

connectionType (entrada)
El tipo de conexión para el cual el objeto ConnectionFactory genera conexiones.

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Devuelve:
El objeto XMSFactoryFactory dedicado al tipo de conexión declarado.

Excepciones:

- NotSupportedException

Propiedades de objetos XMS

Esta sección documenta las propiedades de objeto definidas por XMS.

Esta sección contiene información sobre los siguientes tipos de objeto:

- "Propiedades de conexión" en la página 2086
- "Propiedades de ConnectionFactory" en la página 2086
- "Propiedades de ConnectionMetaData" en la página 2092
- "Propiedades de Destination" en la página 2092
- "Propiedades de InitialContext" en la página 2094
- "Propiedades de Message" en la página 2095
- "Propiedades de MessageConsumer" en la página 2100
- "Propiedades de MessageProducer" en la página 2100
- "Propiedades de sesión" en la página 2100

La descripción de cada tipo de objeto lista las propiedades de un objeto del tipo especificado y proporciona una breve descripción de cada propiedad.

Esta sección también proporciona una definición de cada propiedad (consulte "Definiciones de propiedad" en la página 2101).

Si una aplicación define sus propias propiedades de los objetos descritos en esta sección, no provoca un error, pero puede provocar resultados imprevisibles.

Nota: Los nombres y valores de propiedad de esta sección se muestran en el formato XMSC.*NAME*, que es el formato utilizado para C y C++. Sin embargo, en .NET, el formato del nombre de propiedad puede ser XMSC.*NAME* o XMSC_*NAME*, en función de cómo lo esté utilizando:

- Si está especificando una propiedad, el nombre de propiedad debe tener el formato XMSC.*NAME* tal como se muestra en el ejemplo siguiente:

  ```
  cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
  ```

- Si está especificando una serie, el nombre de propiedad debe tener el formato XMSC_*NAME* tal como se muestra en el ejemplo siguiente:

  ```
  cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
  ```

  En .NET, los nombres y valores de propiedad se proporcionan como constantes en la clase XMSC. Estas constantes identifican series y las utilizaría cualquier aplicación de XMS .NET . Si está utilizando estas constantes predefinidas. los nombres y valores de propiedad tienen el formato XMSC.*NAME*, de modo que, por ejemplo, utilizaría XMSC.USERID, en lugar de XMSC_USERID.

Los tipos de datos también están en el formato utilizado para C/C++. Puede encontrar los valores correspondientes para .NET en Tipos de datos para .NET.

Propiedades de conexión

Una descripción general de las propiedades del objeto Connection, con enlaces a información de referencia más detallada.

Tabla 872. Propiedades de conexión

Nombre de la propiedad	Descripción
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER” en la página 2135	Esta propiedad se utiliza para obtener el nombre del gestor de colas que está conectado.
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID” en la página 2135	Esta propiedad se llena con el ID del gestor de colas después de la conexión.
XMSC_WPM_CONNECTION_PROTOCOL	El protocolo de comunicaciones utilizado para la conexión al motor de mensajería. Esta propiedad es de solo lectura.
XMSC_WPM_HOST_NAME	El nombre de host o la dirección IP del sistema que contiene el motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.
XMSC_WPM_ME_NAME	El nombre del motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.
XMSC_WPM_PORT	El número del puerto en el que estaba a la escucha el motor de mensajería al que está conectada la aplicación. Esta propiedad es de solo lectura.

Un objeto Connection también tiene prioridad de solo lectura que se han derivado de las propiedades de la fábrica de conexiones que se utilizó para crear la conexión. Estas propiedades no solo se han derivado de las propiedades de fábrica de conexiones que se establecieron en el momento cuando se creó la conexión, sino que también de los valores predeterminados de las propiedades que no se establecieron. Las propiedades solo incluyen las que son relevantes para el tipo de servidor de mensajería al que está conectada la aplicación. Los nombres de las propiedades son las mismos que los nombres de las propiedades de fábrica de conexiones.

Propiedades de ConnectionFactory

Una descripción general de las propiedades del objeto ConnectionFactory, con enlaces a información de referencia más detallada.

Tabla 873. Propiedades de ConnectionFactory

Nombre de la propiedad	Descripción
“XMSC_ASYNC_EXCEPTIONS” en la página 2111	Esta propiedad determina si XMS informa de un ExceptionListener solo cuando se interrumpe una conexión, o cuando se produce cualquier excepción de forma asíncrona en una llamada de la API XMS. Esta propiedad se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.
XMSC_CLIENT_ID	El identificador de cliente para una conexión.
XMSC_CONNECTION_TYPE	El tipo del servidor de mensajería al que se conecta una aplicación.
XMSC_PASSWORD	Una contraseña que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
“XMSC_RTT_BROKER_PING_INTERVAL” en la página 2116	El intervalo de tiempo, en milisegundos, transcurrido el cual XMS .NET comprueba la conexión con un servidor de mensajería en tiempo real para detectar cualquier actividad.
XMSC_RTT_CONNECTION_PROTOCOL	El protocolo de comunicación utilizado para una conexión en tiempo real con un intermediario.
XMSC_RTT_HOST_NAME	El nombre de host o la dirección IP del sistema en el que se ejecuta un intermediario.
XMSC_RTT_LOCAL_ADDRESS	El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para una conexión en tiempo real con un intermediario.
XMSC_RTT_MULTICAST	El valor de multidifusión para una fábrica de conexiones o destino.
XMSC_RTT_PORT	El número del puerto en el cual un intermediario escucha solicitudes entrantes.
XMSC_USERID	Un identificador de usuario que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería.
XMSC_WMQ_BROKER_CONTROLQ	Nombre de la cola de control utilizada por un intermediario. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_BROKER_PUBQ	El nombre de la cola supervisada por un intermediario donde las aplicaciones envían mensajes que publican. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_BROKER_QMGR	El nombre del gestor de colas al que está conectado un intermediario. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
XMSC_WMQ_BROKER_SUBQ	El nombre de la cola de suscriptor para un consumidor de mensajes no duraderos. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_BROKER_VERSION	El tipo de intermediario utilizado por la aplicación para una conexión o para el destino. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
“XMSC_WMQ_CCDTURL” en la página 2121	Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de la definición de canal de cliente y, también, especifica cómo se puede acceder al archivo.
XMSC_WMQ_CHANNEL	El nombre del canal que se va a utilizar para una conexión.
“XMSC_WMQ_CLIENT_RECONNECT_OPTIONS” en la página 2122	Esta propiedad especifica las opciones de reconexión de cliente para nuevas conexiones creadas por esta fábrica
“XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT” en la página 2122	Esta propiedad especifica la duración de tiempo, en segundos, que una conexión de cliente intenta reconectarse.
XMSC_WMQ_CONNECTION_MODE	La modalidad a través de la cual una aplicación se conecta a un gestor de colas.
“XMSC_WMQ_CONNECTION_NAME_LIST” en la página 2123	Esta propiedad especifica los hosts a los que el cliente intenta reconectarse después de que se hayan interrumpido sus conexiones.
XMSC_WMQ_FAIL_IF_QUIESCE	Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.
XMSC_WMQ_HOST_NAME	El nombre de host o la dirección IP del sistema en el cual se ejecuta un gestor de colas.
XMSC_WMQ_LOCAL_ADDRESS	Para una conexión a un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se van a utilizar, o ambos.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
XMSC_WMQ_MESSAGE_SELECTION	Determina si la selección de mensajes la realiza el cliente XMS o el intermediario. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_MSG_BATCH_SIZE	El número máximo de mensajes que se va a recuperar de una cola en un lote cuando se utiliza la entrega de mensajes asíncrona. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_POLLING_INTERVAL	Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este valor es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
“XMSC_WMQ_PROVIDER_VERSION” en la página 2132	La versión, release, nivel de modificación y fixpack del gestor de colas al que tiene intención conectarse la aplicación.
XMSC_WMQ_PORT	El número del puerto en el cual un gestor de colas escucha solicitudes entrantes.
XMSC_WMQ_PUB_ACK_INTERVAL	Número de mensajes publicados por un publicador antes de que el cliente de XMS solicite un acuse de recibo al intermediario. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
“XMSC_WMQ_PUT_ASYNC_ALLOWED” en la página 2127	Esta propiedad determina si los productores de mensajes están permitidos para utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.
XMSC_WMQ_QMGR_CCSID	El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se intercambian los campos de datos de caracteres definidos en la interfaz de cola de mensajes (MQI) entre el cliente de XMS y el cliente de IBM MQ .
XMSC_WMQ_QUEUE_MANAGER	El nombre del gestor de colas al que conectarse.
XMSC_WMQ_RECEIVE_EXIT	Identifica una salida de recepción de canal que se va a ejecutar.
XMSC_WMQ_RECEIVE_EXIT_INIT	Los datos de usuario que se pasan a una salida de recepción de canal cuando se llama.
XMSC_WMQ_SECURITY_EXIT	Identifica una salida de seguridad de canal.
XMSC_WMQ_SECURITY_EXIT_INIT	Los datos de usuario que se pasan a una salida de seguridad de canal cuando se llama.
“XMSC_WMQ_SEND_CHECK_COUNT” en la página 2137	El número de llamadas de envío que se van a permitir entre las comprobaciones de errores de colocación asíncrona dentro de una sesión XMS única sin transacción.
XMSC_WMQ_SEND_EXIT	Identifica una salida de emisión de canal.
XMSC_WMQ_SEND_EXIT_INIT	Los datos de usuario que se pasan a las salidas de emisión de canal cuando se llaman.
“XMSC_WMQ_SHARE_CONV_ALLOWED” en la página 2137	Indica si una conexión de cliente puede compartir su socket con otras conexiones XMS de nivel superior del mismo proceso con el mismo gestor de colas, si las definiciones de canal coinciden. Esta propiedad se proporciona para permitir un aislamiento completo de conexiones en sockets separados si es necesario para el desarrollo de aplicaciones, el mantenimiento o por motivos operativos.
XMSC_WMQ_SSL_CERT_STORES	Las ubicaciones de los servidores que contienen las listas de revocaciones de certificados (CRL) que se van a utilizar en una conexión SSL a un gestor de colas.
XMSC_WMQ_SSL_CIPHER_SPEC	El nombre de la CipherSpec que se va a utilizar en una conexión segura a un gestor de colas.
XMSC_WMQ_SSL_CIPHER_SUITE	El nombre de la CipherSuite que se va a utilizar en una conexión TLS a un gestor de colas. El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.
XMSC_WMQ_SSL_CRYPTO_HW	Detalles de configuración para el hardware de cifrado conectado al sistema cliente.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
XMSC_WMQ_SSL_FIPS_REQUIRED	El valor de esta propiedad determina si una aplicación puede o no puede utilizar suites de cifrado no compatibles con FIPS. Si esta propiedad se establece en true (verdadero), solo se utilizan algoritmos FIPS para la conexión cliente/servidor.
XMSC_WMQ_SSL_KEY_REPOSITORY	La ubicación de un archivo de base de datos de claves en el cual se almacenan claves y certificados.
XMSC_WMQ_SSL_KEY_RESETCOUNT	El KeyResetCount representa el número total de bytes sin cifrado enviados y recibidos en una conversación SSL antes de renegociar la clave secreta.
XMSC_WMQ_SSL_PEER_NAME	El nombre de igual que se va a utilizar en una conexión SSL a un gestor de colas.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Indica si se deben recuperar todos los mensajes de colas dentro del control de punto de sincronización.
“XMSC_WMQ_TARGET_CLIENT” en la página 2143	
XMSC_WMQ_TEMP_Q_PREFIX	El prefijo utilizado para formar el nombre de la cola dinámica de IBM MQ que se crea cuando la aplicación crea una cola temporal de XMS .
XMSC_WMQ_TEMP_TOPIC_PREFIX	Al crear temas temporales, XMS genera una serie de tema con el formato "`TEMP/TEMPTOPICPREFIX/unique_id`", o si esta propiedad contiene el valor predeterminado, se genera esta serie, "`TEMP/unique_id`". Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.
XMSC_WMQ_TEMPORARY_MODEL	Nombre de la cola modelo de IBM MQ a partir de la cual se crea una cola dinámica cuando la aplicación crea una cola temporal de XMS .
XMSC_WPM_BUS_NAME	Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.
XMSC_WPM_CONNECTION_PROXIMITY	El valor de proximidad de conexión para la conexión.
XMSC_WPM_DUR_SUB_HOME	El nombre del motor de mensajería donde se gestionan todas las suscripciones duraderas para una conexión o destino.
XMSC_WPM_LOCAL_ADDRESS	Para una conexión a un bus de integración de servicios, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar, o ambos.
XMSC_WPM_NON_PERSISTENT_MAP	El nivel de fiabilidad de los mensajes no persistente que se envían utilizando la conexión.

Tabla 873. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	**Descripción**
XMSC_WPM_PERSISTENT_MAP	El nivel de fiabilidad de los mensajes persistentes que se envían mediante la conexión.
XMSC_WPM_PROVIDER_ENDPOINTS	Una secuencia de una o más direcciones de punto final de servidores de programa de arranque.
XMSC_WPM_TARGET_GROUP	El nombre de un grupo de destino de motores de mensajería.
XMSC_WPM_TARGET_SIGNIFICANCE	La importancia del grupo de destinos de motores de mensajería.
XMSC_WPM_TARGET_TRANSPORT_CHAIN	El nombre de la cadena de transporte de entrada que debe utilizar la aplicación para conectarse a un motor de mensajería.
XMSC_WPM_TARGET_TYPE	El tipo del grupo de destinos de motores de mensajería.
XMSC_WPM_TEMP_Q_PREFIX	El prefijo utilizado para formar el nombre de la cola temporal que se crea en el bus de integración de servicios cuando la aplicación crea una cola temporal de XMS .
XMSC_WPM_TEMP_TOPIC_PREFIX	El prefijo utilizado para formar el nombre de un tema temporal que crea la aplicación.

Propiedades de ConnectionMetaData

Una descripción general de las propiedades del objeto ConnectionMetaData, con enlaces a información de referencia más detallada.

Tabla 874. Propiedades de ConnectionMetaData	
Nombre de la propiedad	**Descripción**
XMSC_JMS_MAJOR_VERSION	El número de versión principal de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.
XMSC_JMS_MINOR_VERSION	El número de versión menor de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.
XMSC_JMS_VERSION	El identificador de versión de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.
XMSC_MAJOR_VERSION	El número de versión del cliente de XMS . Esta propiedad es de solo lectura.
XMSC_MINOR_VERSION	El número de release del cliente de XMS . Esta propiedad es de solo lectura.
XMSC_PROVIDER_NAME	El proveedor del cliente de XMS . Esta propiedad es de solo lectura.
XMSC_VERSION	El identificador de versión de la cliXMSent. Esta propiedad es de sólo lectura.

Propiedades de Destination

Una descripción general de las propiedades del objeto Destination, con enlaces a información de referencia a más detallada.

Tabla 875. Propiedades de Destination	
Nombre de la propiedad	**Descripción**
XMSC_DELIVERY_MODE	La modalidad de entrega de mensajes enviados al destino.
XMSC_PRIORITY	La prioridad de los mensajes enviados al destino.
XMSC_RTT_MULTICAST	El valor de multidifusión para una fábrica de conexiones o destino.
XMSC_TIME_TO_LIVE	El tiempo de vida para los mensajes enviados al destino.
XMSC_WMQ_BROKER_VERSION	El tipo de intermediario utilizado por la aplicación para una conexión o para el destino.
XMSC_WMQ_CCSID	El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se encuentran las series de datos de caracteres en el cuerpo de un mensaje cuando el cliente XMS reenvía el mensaje al destino.
XMSC_WMQ_DUR_SUBQ	El nombre de la cola de suscriptor para un suscriptor duradero que está recibiendo mensajes del destino. **Nota:** Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_ENCODING	Cómo se representan los datos numéricos en el cuerpo de un mensaje cuando el cliente XMS reenvía el mensaje al destino.
XMSC_WMQ_FAIL_IF_QUIESCE	Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.
“XMSC_WMQ_MESSAGE_BODY” en la página 2125	Esta propiedad determina si una aplicación XMS procesa la MQRFH2 de un mensaje IBM MQ como parte de la carga útil del mensaje (es decir, como parte del cuerpo del mensaje).
“XMSC_WMQ_MQMD_MESSAGE_CONTEXT” en la página 2126	Determina qué nivel de contexto de mensaje debe establecer la aplicación XMS . La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.
“XMSC_WMQ_MQMD_READ_ENABLED” en la página 2127	Esta propiedad determina si una aplicación XMS puede extraer o no los valores de los campos MQMD.
“XMSC_WMQ_MQMD_WRITE_ENABLED” en la página 2127	Esta propiedad determina si una aplicación XMS puede establecer los valores de los campos MQMD o no.
“XMSC_WMQ_READ_AHEAD_ALLOWED” en la página 2128	Esta propiedad determina si los consumidores de mensajes y los navegadores de colas están autorizados para utilizar la lectura anticipada para obtener mensajes no persistentes y no transaccionales de este destino en un almacenamiento intermedio interno antes de recibirlos.

Tabla 875. Propiedades de Destination (continuación)	
Nombre de la propiedad	**Descripción**
“XMSC_WMQ_READ_AHEAD_CLOSE_POLICY” en la página 2129	Esta propiedad determina, para los mensajes que se están entregando a un escucha de mensajes asíncronos, qué sucede con los mensajes en el almacenamiento de lectura anticipada interno, cuando se cierra el consumidor de mensajes.
“XMSC_WMQ_RECEIVE_CCSID” en la página 2134	La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora, a menos que XMSC_WMQ_RECEIVE_CONVERSION se establezca en WMQ_RECEIVE_CONVERSION_QMGR.
“XMSC_WMQ_RECEIVE_CONVERSION” en la página 2134	La propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.
XMSC_WMQ_TARGET_CLIENT	Indica si los mensajes enviados al destino contienen una cabecera MQRFH2.
XMSC_WMQ_TEMP_TOPIC_PREFIX	Al crear temas temporales, XMS genera una serie de tema con el formato "`TEMP/TEMPTOPICPREFIX/unique_id`", o si esta propiedad contiene el valor predeterminado, se genera esta serie, "`TEMP/unique_id`". Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.
XMSC_WPM_BUS_NAME	Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.
XMSC_WPM_TOPIC_SPACE	El nombre del espacio de tema que contiene el tema.

Propiedades de InitialContext

Una descripción general de las propiedades del objeto InitialContext, con enlaces a información de referencia más detallada.

Tabla 876. Propiedades de InitialContext	
Nombre de la propiedad	**Descripción**
XMSC_IC_PROVIDER_URL	Se utiliza para localizar el directorio de denominación JNDI de forma que no es necesario que el servicio de denominación COS esté en el mismo servidor que el servicio web.
XMSC_IC_SECURITY_AUTHENTICATION	Basado en la interfaz de contexto Java SECURITY_AUTHENTICATION. Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_SECURITY_CREDENTIALS	Basado en la interfaz de contexto Java SECURITY_CREDENTIALS. Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_SECURITY_PRINCIPAL	Basado en la interfaz de contexto Java SECURITY_PRINCIPAL. Esta propiedad solo es aplicable al contexto de denominación COS.

Tabla 876. Propiedades de InitialContext (continuación)	
Nombre de la propiedad	**Descripción**
XMSC_IC_SECURITY_PROTOCOL	Basado en la interfaz de contexto Java SECURITY_PROTOCOL Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_URL	Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados. Para contextos de denominación COS, la dirección del servicio web que busca los objetos en el directorio.

Propiedades de Message

Una descripción general de las propiedades del objeto Message, con enlaces a información de referencia más detallada.

Tabla 877. Propiedades de Message	
Nombre de la propiedad	**Descripción**
JMS_IBM_CHARACTER_SET	El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se encuentran las series de datos de caracteres en el cuerpo del mensaje cuando el cliente de XMS reenvía el mensaje a su destino previsto. En XMS, esta propiedad tiene un valor numérico y se correlaciona con el identificador de juego de caracteres codificados (CCSID). Si embargo, esta propiedad se basa en una propiedad JMS de modo que tiene un valor de tipo serie y se correlaciona con el juego de caracteres Java que representa este CCSID numérico.
JMS_IBM_ENCODING	Cómo se representan los datos numéricos en el cuerpo del mensaje cuando el cliente XMS reenvía el mensaje a su destino previsto.
JMS_IBM_EXCEPTIONMESSAGE	Texto que describe la razón por la cual el mensaje se ha enviado al destino de excepciones. Esta propiedad es de solo lectura.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	El nombre del destino en el que estaba el mensaje antes de que se enviara al destino de excepciones.
JMS_IBM_EXCEPTIONREASON	Un código de razón que indica la razón por la cual el mensaje se envió al destino de excepciones.
JMS_IBM_EXCEPTIONTIMESTAMP	La hora cuando se envió el mensaje al destino de excepciones.
JMS_IBM_FEEDBACK	Un código que indica la naturaleza de un mensaje de informe.
JMS_IBM_FORMAT	La naturaleza de los datos de aplicación en el mensaje.
JMS_IBM_LAST_MSG_IN_GROUP	Indica si el mensaje es el último mensaje de un grupo de mensajes.
JMS_IBM_MSGTYPE	El tipo del mensaje.
JMS_IBM_PUTAPPLTYPE	El tipo de la aplicación que envió el mensaje.
JMS_IBM_PUTDATE	La fecha cuando se envió el mensaje.

Tabla 877. Propiedades de Message (continuación)	
Nombre de la propiedad	**Descripción**
JMS_IBM_PUTTIME	La hora cuando se envió el mensaje.
JMS_IBM_REPORT_COA	Solicitar los mensajes de informe 'confirmar en llegada', especificando cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_COD	Solicitar mensaje de informe 'confirmar en llegada', que especifica cuántos datos d e aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_DISCARD_MSG	Solicitar que el mensaje se descarte si no se puede entregar a su destino previsto.
JMS_IBM_REPORT_EXCEPTION	Solicitar mensajes de informe de excepción, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_EXPIRATION	Solicitar mensajes de informe de caducidad, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_NAN	Solicitar mensajes de informe de notificación de acción negativa.
JMS_IBM_REPORT_PAN	Solicitar mensajes de informe de notificación de acción positiva.
JMS_IBM_REPORT_PASS_CORREL_ID	Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.
JMS_IBM_REPORT_PASS_MSG_ID	Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.
JMS_IBM_RETAIN	Establecer esta propiedad indica al gestor de colas que trate un mensaje como una Publicación retenida.
JMS_IBM_SYSTEM_MESSAGEID	Un identificador que identifica el mensaje de forma exclusiva dentro del bus de integración de servicios. Esta propiedad es de solo lectura.
JMSX_APPID	El nombre de la aplicación que envió el mensaje.
JMSX_DELIVERY_COUNT	El número de intentos para entregar el mensaje.
JMSX_GROUPID	El identificador del grupo de mensajes al que pertenece el mensaje.
JMSX_GROUPSEQ	El número de secuencia del mensaje de un grupo de mensajes.
JMSX_USERID	El identificador de usuario asociado con la aplicación que envió el mensaje.

Propiedades JMS_IBM_MQMD*

IBM Message Service Client for .NET permite a las aplicaciones cliente leer/escribir campos MQMD utilizando distintas API. También permite el acceso a datos de mensaje MQ. De forma predeterminada, el acceso a MQMD está inhabilitado y la aplicación la debe habilitar de forma explícita utilizando las

propiedades Destination XMSC_WMQ_MQMD_WRITE_ENABLED y XMSC_WMQ_MQMD_READ_ENABLED. Estas dos propiedades son independientes entre sí.

Todos los campos MQMD excepto StrucId y Version se exponen como propiedades de objeto Message adicionales y se añaden como prefijo a JMS_IBM_MQMD.

Las propiedades JMS_IBM_MQMD* tienen una mayor prioridad sobre otras propiedades como JMS_IBM* descritas en la tabla anterior.

Envío de mensajes

Están representados todos los campos MQMD excepto StrucId y Version. Estas propiedades solo hacen referencia a los campos MQMD; donde una propiedad se produce tanto en MQMD como en la cabeceraMQRFH2, la versión en MQRFH2 no se establece ni se extrae. Se puede establecer cualquiera de estas propiedades, excepto JMS_IBM_MQMD_BackoutCount. Se ignora cualquier valor establecido para JMS_IBM_MQMD_BackoutCount.

Si una propiedad tiene una longitud máxima y proporciona un valor que es demasiado largo, el valor se trunca.

Para determinadas propiedades, también debe establecer la propiedad XMSC_WMQ_MQMD_MESSAGE_CONTEXT en el objeto Destination. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto. Si no establece XMSC_WMQ_MQMD_MESSAGE_CONTEXT en un valor apropiado, se ignora el valor de propiedad. Si establece XMSC_WMQ_MQMD_MESSAGE_CONTEXT en un valor apropiado, pero no tiene suficiente autoridad de contexto para el gestor de colas, se emite una excepción. Las propiedades que requieren valores específicos de XMSC_WMQ_MQMD_MESSAGE_CONTEXT son las siguientes.

Las propiedades siguientes requieren que XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecido en XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT o XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

Las propiedades siguientes requieren que XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecido en XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

Recepción de mensajes

Todas estas propiedades están disponibles en un mensaje recibido si la propiedad XMSC_WMQ_MQMD_READ_ENABLED está establecida en true, independientemente de las propiedades reales que ha establecido la aplicación productora. Una aplicación no puede modificar las propiedades de un mensaje recibido, a menos que primero se borren todas las propiedades, de acuerdo con la especificación JMS. El mensaje recibido se puede enviar sin modificar las propiedades.

Nota: Si la aplicación recibe un mensaje de un destino con la propiedad XMSC_WMQ_MQMD_READ_ENABLED establecida en true, y lo envía a un destino con XMSC_WMQ_MQMD_WRITE_ENABLED establecido en true, esto genera que todos los valores de campo MQMD del mensaje recibido se copien en el mensaje enviado. Tabla de propiedades

Tabla 878. Propiedades del objeto Message que representa los campos MQMD		
Propiedad	**Descripción**	**Tipo**
JMS_IBM_MQMD_REPORT	Opciones para mensajes de informe	System.Int32
JMS_IBM_MQMD_MSGTYPE	Tipo de mensaje	System.Int32
JMS_IBM_MQMD_EXPIRY	Duración del mensaje	System.Int32
JMS_IBM_MQMD_FEEDBACK	Código de comentario o razón	System.Int32
JMS_IBM_MQMD_ENCODING	Codificación numérica de datos de mensaje	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	Identificador de juego de caracteres de datos de mensaje	System.Int32
JMS_IBM_MQMD_FORMAT	Nombre de formato de datos de mensaje	System.String
JMS_IBM_MQMD_PRIORITY **Nota:** Si asigna un valor a JMS_IBM_MQMD_PRIORITY que no está dentro del rango de 0 a 9, este valor infringe la especificación JMS.	Prioridad de mensaje	System.Int32
JMS_IBM_MQMD_PERSISTENCE	Persistencia de los mensajes	System.Int32
JMS_IBM_MQMD_MSGID **Nota:** La especificación JMS indica que el ID de mensaje debe ser establecido por el proveedor JMS y que se debe ser exclusivo o nulo. Si asigna un valor a JMS_IBM_MQMD_MSGID, este valor se copia en el JMSMessageID. De esta forma, el proveedor JMS no lo establece y podría no ser exclusivo: este valor infringe la especificación JMS.	Identificador del mensaje	Matriz de bytes **Nota:** El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_CORRELID **Nota:** Si asigna un valor a JMS_IBM_MQMD_CORRELID que empieza con la serie 'ID:', este valor infringe la especificación JMS.	Identificador de correlación	Matriz de bytes **Nota:** El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_BACKOUTCOUNT	Contador de restitución	System.Int32
JMS_IBM_MQMD_REPLYTOQ	Nombre de la cola de respuestas	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Nombre del gestor de colas de respuestas	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Identificador de usuario	System.String

Tabla 878. Propiedades del objeto Message que representa los campos MQMD (continuación)		
Propiedad	**Descripción**	**Tipo**
JMS_IBM_MQMD_ACCOUNTINGTOKEN	Señal de contabilidad	Matriz de bytes **Nota:** El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_APPLIDENTITYDATA	Datos de aplicación relacionados con la identidad	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	Tipo de aplicación que coloca el mensaje	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	Nombre de la aplicación que coloca el mensaje	System.String
JMS_IBM_MQMD_PUTDATE	Fecha cuando se colocó el mensaje	System.String
JMS_IBM_MQMD_PUTTIME	Hora cuando se colocó el mensaje	System.String
JMS_IBM_MQMD_APPLORIGINDATA	Datos de aplicación relacionados con el origen	System.String
JMS_IBM_MQMD_GROUPID	Identificador de grupo	Matriz de bytes **Nota:** El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_MSGSEQNUMBER	Número de secuencia del mensaje local dentro del grupo	System.Int32
JMS_IBM_MQMD_OFFSET	Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Distintivos de mensajes	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Longitud del mensaje original	System.Int32

Consulte MQMD si desea más detalles.

Ejemplos

Este ejemplo provoca que un mensaje se coloque en una cola o un tema con MQMD.UserIdentifier establecido en"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
  // ...
```

Es necesario establecer XMSC_WMQ_MQMD_MESSAGE_CONTEXT antes de establecer JMS_IBM_MQMD_USERIDENTIFIER. Si desea más información sobre el uso de XMSC_WMQ_MQMD_MESSAGE_CONTEXT, consulte las propiedades del objeto Message.

De forma similar, puede extraer el contenido de los campos MQMD estableciendo XMSC_WMQ_MQMD_READ_ENABLED en true antes de recibir un mensaje y, después, utilizar los métodos get del mensaje como, por ejemplo, getStringProperty. Las propiedades recibidas son de solo lectura.

Este ejemplo tiene como resultado que el campo de valor que contiene el valor del campo MQMD.ApplIdentityData de un mensaje se obtenga de una cola o de un tema.

```
  // 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);
```

Propiedades de MessageConsumer

Una descripción general de las propiedades del objeto MessageConsumer, con enlaces a información de referencia más detallada.

Tabla 879. Propiedades de MessageConsumer

Nombre de la propiedad	Descripción
XMSC_IS_SUBSCRIPTION_MULTICAST	Indica si los mensajes se están entregando al consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Esta propiedad es de solo lectura.
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	Indica si los mensajes se entregan al consumidor de mensajes utilizando WebSphere MQ Multicast Transport con una calidad de servicio fiable. Esta propiedad es de solo lectura.

Consulte las Propiedades .NET de IMessageConsumer si desea más detalles.

Propiedades de MessageProducer

Una descripción general de las propiedades del objeto MessageProducer, con enlaces a información de referencia más detallada.

Consulte .Propiedades NET de IMessageProducer para obtener más detalles.

Propiedades de sesión

Una descripción general de las propiedades del objeto Session, con enlaces a información de referencia más detallada.

Consulte .Propiedades NET de ISession para obtener más detalles.

Definiciones de propiedad

Esta sección proporciona una definición de cada propiedad de objeto.

Cada definición de propiedad incluye la información siguiente:

- El tipo de datos de la propiedad
- Los tipos de objeto que tienen la propiedad
- Para una propiedad de Destination, el nombre que se puede utilizar en un identificador uniforme de recursos (URI)
- Una descripción más detallada de la propiedad
- Los valores válidos de la propiedad
- El valor predeterminado de la propiedad

Las propiedades cuyos nombres empiezan con uno de los prefijos siguientes solo son relevantes para el tipo de conexión especificado:

XMSC_RTT
: Las propiedades solo son relevantes para una conexión en tiempo real con un intermediario. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_rtt.h`.

XMSC_WMQ
: Las propiedades sólo son relevantes cuando una aplicación se conecta a un gestor de colas IBM MQ . Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_wmq.h`.

XMSC_WPM
: Las propiedades solo son relevantes cuando una aplicación se conecta a un bus de integración de servicios WebSphere. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_wpm.h`.

A menos que se indique lo contrario en sus definiciones, las propiedades restantes son relevantes para todos los tipos de conexión. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc.h`. Las propiedades cuyos nombres empiezan con el prefijo JMSX son propiedades definidas de JMS de un mensaje, y las propiedades cuyos nombres empiezan con el prefijo JMS_IBM son las propiedades definidas de IBM de un mensaje. Para obtener más información sobre las propiedades de los mensajes, consulte Propiedades de un mensaje de XMS.

A menos que se indique lo contrario en su definición, cada propiedad es relevante en los dominios punto a punto y de publicación/suscripción.

Una aplicación puede obtener y establecer el valor de cualquier propiedad, a menos que la propiedad se haya designado como de solo lectura.

JMS_IBM_CHARACTER_SET

Tipo de datos:
: System.Int32

Propiedad de:
: Mensaje

El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se encuentran las series de datos de caracteres en el cuerpo del mensaje cuando el cliente de XMS reenvía el mensaje a su destino previsto. En XMS, esta propiedad tiene un valor numérico y se correlaciona con el identificador de juego de caracteres codificados (CCSID). Si embargo, esta propiedad se basa en una propiedad JMS de modo que tiene un valor de tipo serie y se correlaciona con el juego de caracteres Java que representa este CCSID numérico. Esta propiedad altera temporalmente cualquier CCSID especificado para el destino mediante la propiedad XMSC_WMQ_CCSID .

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_ENCODING

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Cómo se representan los datos numéricos en el cuerpo del mensaje cuando el cliente XMS reenvía el mensaje a su destino previsto. Esta propiedad altera temporalmente cualquier codificación especificada para el destino mediante la propiedad XMSC_WMQ_ENCODING . La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **Encoding** de un descriptor de mensaje.

Una aplicación puede utilizar las constantes con nombre siguientes para establecer la propiedad:

Constante con nombre	Significado
MQENC_INTEGER_NORMAL	Codificación de entero normal
MQENC_INTEGER_REVERSED	Codificación de entero inverso
MQENC_DECIMAL_NORMAL	Codificación de decimal empaquetado normal
MQENC_DECIMAL_REVERSED	Codificación de decimal empaquetado inverso
MQENC_FLOAT_IEEE_NORMAL	Codificación de coma flotante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codificación de coma flotante IEEE inversa
MQENC_FLOAT_S390	z/OS codificación de coma flotante de arquitectura
MQENC_NATIVE	Codificación de máquina nativa

Para formar un valor para la propiedad, la aplicación puede añadir tres de estas constantes del modo siguiente:

- Una constante cuyo nombre empieza con MQENC_INTEGER, para especificar la representación de enteros binarios
- Una constante cuyo nombre empieza con MQENC_DECIMAL, para especificar la representación de enteros decimales empaquetados
- Una constante cuyo nombre empieza con MQENC_FLOAT, para especificar la representación de números de coma flotante

De forma alternativa, la aplicación puede establecer la propiedad en MQENC_NATIVE, cuyo valor depende del entorno.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_EXCEPTIONMESSAGE

Tipo de datos:
Cadena

Propiedad de:
Mensaje

Texto que describe la razón por la cual el mensaje se ha enviado al destino de excepciones. Esta propiedad es de solo lectura.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Tipo de datos:
Cadena

Propiedad de:
Mensaje

El nombre del destino en el que estaba el mensaje antes de que se enviara al destino de excepciones.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONREASON

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Un código de razón que indica la razón por la cual el mensaje se envió al destino de excepciones.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONTIMESTAMP

Tipo de datos:
System.Int64

Propiedad de:
Mensaje

La hora cuando se envió el mensaje al destino de excepciones.

La hora se expresa en milisegundos desde 00:00:00 GMT del 1 de enero de 1970.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_FEEDBACK

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Un código que indica la naturaleza de un mensaje de informe.

Los valores válidos de la propiedad son los códigos de comentarios (feedback) y códigos de razón que se pueden especificar en el campo **Feedback** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_FORMAT

Tipo de datos:
Cadena

Propiedad de:
Mensaje

La naturaleza de los datos de aplicación en el mensaje.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **Format** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_LAST_MSG_IN_GROUP

Tipo de datos:
System.Boolean

Propiedad de:
Mensaje

Indica si el mensaje es el último mensaje de un grupo de mensajes.

Establezca la propiedad en true si el mensaje es el último mensaje de un grupo de mensajes. De lo contrario, establezca la propiedad en false, o no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor true corresponde al distintivo de estado MQMF_LAST_MSG_IN_GROUP, que se puede especificar en el campo **MsgFlags** de un descriptor de mensaje.

Esta propiedad se ignora en el dominio de publicación/suscripción y no es relevante cuando una aplicación se conecta a un bus de integración de servicios.

JMS_IBM_MSGTYPE

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

El tipo del mensaje.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQMT_DATAGRAM	El mensaje es uno que no requiere una respuesta.
MQMT_REQUEST	El mensaje es uno que requiere una respuesta.
MQMT_REPLY	El mensaje es un mensaje de respuesta.
MQMT_REPORT	El mensaje e sun mensaje de informe.

Estos valores corresponden a los tipos de mensaje que se pueden especificar en el campo de **MsgType** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTAPPLTYPE

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

El tipo de la aplicación que envió el mensaje.

Los valores válidos de la propiedad son los tipos de aplicación que se pueden especificar en el campo **PutApplType** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTDATE

Tipo de datos:
Cadena

Propiedad de:
Mensaje

La fecha cuando se envió el mensaje.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **PutDate** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTTIME

Tipo de datos:
Cadena

Propiedad de:
Mensaje

La hora cuando se envió el mensaje.

Los valores válidos de la propiedad son los mismo que los valores que se pueden especificar en el campo **PutTime** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_REPORT_COA

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar los mensajes de informe 'confirmar en llegada', especificando cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_COA	Solicitar mensajes de informe 'confirmar en llegada', si datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_COA_WITH_DATA	Solicitar mensajes de informe 'confirmar en llegada', con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_COA_WITH_FULL_DATA	Solicitar mensajes de informe 'confirmar en llegada', con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje. Si desea más información sobre estas opciones, consulte Informe (MQLONG).

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_COD

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar mensaje de informe 'confirmar en llegada', que especifica cuántos datos d e aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_COD	Solicitar mensajes de informe 'confirmar en llegada', con ninguno de los datos de aplicación del mensaje original incluido en un mensaje de informe.
MQRO_COD_WITH_DATA	Solicitar mensajes de informe 'confirmar en llegada', con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_COD_WITH_FULL_DATA	Solicitar mensajes de informe 'confirmar en llegada', con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_DISCARD_MSG

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar que el mensaje se descarte si no se puede entregar a su destino previsto.

Establezca la propiedad en MQRO_DISCARD_MSG para solicitar que el mensaje se descarte si no se puede entregar a su destino previsto. Si necesita que el mensaje se coloque en una cola de mensajes no entregados en su lugar, o que se envíe a un destino de excepciones, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_DISCARD_MSG corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_EXCEPTION

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar mensajes de informe de excepción, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_EXCEPTION	Solicitar mensajes de informe de excepción, sin ningún dato de aplicación del mensajeoriginal incluido en un mensaje de informe.
MQRO_EXCEPTION_WITH_DATA	Solicitar mensajes de informe de excepción, con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_EXCEPTION_WITH_FULL_DATA	Solicitar mensajes de informe de excepción, con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_EXPIRATION

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar mensajes de informe de caducidad, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_EXPIRATION	Solicitar mensajes de informe de caducidad, sin ninguno de los datos de aplicación del mensaje original incluido en un mensaje de informe.
MQRO_EXPIRATION_WITH_DATA	Solicitar mensajes de informe de caducidad, con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_EXPIRATION_WITH_FULL_DATA	Solicitar mensajes de informe de caducidad, con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_NAN

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar mensajes de informe de notificación de acción negativa.

Establezca la propiedad en MQRO_NAN para solicitar mensajes de informe de notificación de acción negativa. Si no necesita mensajes de informe de notificación de acción negativa, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_NAN corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_PAN

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicitar mensajes de informe de notificación de acción positiva.

Establezca la propiedad en MQRO_PAN para solicitar mensajes de informe de notificación de acción positiva. Si no necesita mensajes de informe de notificación de acción positiva, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_PAN corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_PASS_CORREL_ID

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_PASS_CORREL_ID	Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.
MQRO_COPY_MSG_ID_TO_CORREL_ID	Solicite que el identificador de correlación de cualquier informe o mensaje de respuesta sea el mismo que el identificador de mensaje del mensaje original.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

El valor predeterminado de la propiedad es MQRO_COPY_MSG_ID_TO_CORREL_ID.

JMS_IBM_REPORT_PASS_MSG_ID

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_PASS_MSG_ID	Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.
MQRO_NEW_MSG_ID	Solicite que se genere un nuevo identificador de mensaje para cada mensaje de informe o respuesta.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo Report de un descriptor de mensaje.

El valor predeterminado de la propiedad es MQRO_NEW_MSG_ID.

JMS_IBM_RETAIN

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

Establecer esta propiedad indica al gestor de colas que trate un mensaje como una Publicación retenida. Cuando un suscriptor recibe mensajes de temas, podría recibir mensajes adicionales inmediatamente después de la suscripción, más allá de los mensajes recibidos en releases anteriores. Estos mensajes son las publicaciones retenidas opcionales para los temas suscritos. Para cada tema que coincida con la suscripción, si hay una publicación retenida, la publicación pasa a estar disponible para la entrega al consumidor de mensajes de la suscripción.

RETAIN_PUBLICATION es el único valor válido para esta propiedad. De forma predeterminada, esta propiedad no está establecida.

Nota: Esta propiedad solo es relevante en el dominio de publicación/suscripción únicamente.

JMS_IBM_SYSTEM_MESSAGEID

Tipo de datos:
Cadena

Propiedad de:
Mensaje

Un identificador que identifica el mensaje de forma exclusiva dentro del bus de integración de servicios. Esta propiedad es de solo lectura.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios.

JMSX_APPID

Tipo de datos:
Cadena

Propiedad de:
Mensaje

El nombre de la aplicación que envió el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre JMS JMSXAppID. Para obtener más información sobre la propiedad, consulte *Java Message Service Specification, Version 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_DELIVERY_COUNT

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

El número de intentos para entregar el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXDeliveryCount. Para obtener más información sobre la propiedad, consulte *Java Message Service Specification, Version 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_GROUPID

Tipo de datos:
Cadena

Propiedad de:
Mensaje

El identificador del grupo de mensajes al que pertenece el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXGroupID. Para obtener más información sobre la propiedad, consulte *Java Message Service Specification, Version 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_GROUPSEQ

Tipo de datos:
System.Int32

Propiedad de:
Mensaje

El número de secuencia del mensaje de un grupo de mensajes.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXGroupSeq. Para obtener más información sobre la propiedad, consulte *Java Message Service Specification, Version 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_USERID

Tipo de datos:
Cadena

Propiedad de:
Mensaje

El identificador de usuario asociado con la aplicación que envió el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXUserID. Para obtener más información sobre la propiedad, consulte *Java Message Service Specification, Version 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

XMSC_ASYNC_EXCEPTIONS

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: ASYNCEXCEPTION

Nombre abreviado de la herramienta de administración JMS: AEX

Esta propiedad determina si XMS informa de un ExceptionListener solo cuando se interrumpe una conexión, o cuando se produce cualquier excepción de forma asíncrona en una llamada de la API XMS. Esta propiedad se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.

Los valores válidos para esta propiedad son:

XMSC_ASYNC_EXCEPTIONS_ALL
Cualquier excepción detectada de forma asíncrona, fuera del ámbito de una llamada a API síncrona, y todas las excepciones de interrupción de conexión se envían al ExceptionListener.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
Solo las excepciones que indican una conexión interrumpida se envían al ExceptionListener. Cualquier otra excepción que se produce durante el proceso asíncrono no se notifica al ExceptionListener y, por lo tanto, la aplicación no está informada de estas excepciones.

De forma predeterminada, esta propiedad está establecida en XMSC_ASYNC_EXCEPTIONS_ALL.

XMSC_CLIENT_ID

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CLIENTID

Nombre abreviado de la herramienta de administración JMS: CID

El identificador de cliente para una conexión.

Un identificador de cliente se utiliza solo para dar soporte a suscripciones duraderas en el dominio de publicación/suscripción y se ignora en el dominio punto a punto. Para obtener más información sobre cómo establecer identificadores de cliente, consulte ConnectionFactories y objetos Connection.

Esta propiedad no es relevante para una conexión en tiempo real con un intermediario.

XMSC_CONNECTION_TYPE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El tipo del servidor de mensajería al que se conecta una aplicación.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_CT_RTT	Una conexión en tiempo real con un intermediario.
XMSC_CT_WMQ	Una conexión con un gestor de colas de IBM MQ .

Valor válido	Significado
XMSC_CT_WPM	Una conexión a un WebSphere Application Server service integration bus.

De forma predeterminada, la propiedad no está establecida.

XMSC_DELIVERY_MODE

Tipo de datos:
System.Int32

Propiedad de:
Destino

Nombre utilizado en un URI:

`persistence` (para un destino IBM MQ)
`deliveryMode` (para un destino de proveedor de mensajería WebSphere predeterminado)

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: PERSISTENCE

Nombre abreviado de la herramienta de administración JMS: PER

La modalidad de entrega de mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_DELIVERY_NOT_PERSISTENT	Un mensaje enviado al destino es no persistente. Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino es una cola IBM MQ , el valor del atributo de cola *`DefPersistence`* también se ignora.
XMSC_DELIVERY_PERSISTENT	Un mensaje enviado al destino es persistente. Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino es una cola IBM MQ , el valor del atributo de cola *`DefPersistence`* también se ignora.
XMSC_DELIVERY_AS_APP	Un mensaje enviado al destino tiene la modalidad de entrega especificada en la llamada Send. Si la llamada Send no especifica ninguna modalidad de entrega, en su lugar, se utiliza la modalidad de entrega predeterminada del productor de mensajes. Si el destino es una cola IBM MQ , el valor del atributo de cola *`DefPersistence`* se ignora.
XMSC_DELIVERY_AS_DEST	Si el destino es una cola IBM MQ , un mensaje colocado en la cola tiene la modalidad de entrega especificada por el valor del atributo de cola *`DefPersistence`*. Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino no es una cola IBM MQ , el significado es el mismo que el de XMSC_DELIVERY_AS_APP.

El valor predeterminado es XMSC_DELIVERY_AS_APP.

XMSC_IC_PROVIDER_URL

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Se utiliza para localizar el directorio de denominación JNDI de forma que no es necesario que el servicio de denominación COS esté en el mismo servidor que el servicio web.

XMSC_IC_SECURITY_AUTHENTICATION

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Basado en la interfaz de contexto Java SECURITY_AUTHENTICATION. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_CREDENTIALS

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Basado en la interfaz de contexto Java SECURITY_CREDENTIALS. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_PRINCIPAL

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Basado en la interfaz de contexto Java SECURITY_PRINCIPAL. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_PROTOCOL

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Basado en la interfaz de contexto Java SECURITY_PROTOCOL Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_URL

Tipo de datos:
Cadena

Propiedad de:
InitialContext

Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados.

Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados.

XMSC_IS_SUBSCRIPTION_MULTICAST

Tipo de datos:
System.Boolean

Propiedad de:
MessageConsumer

Indica si los mensajes se están entregando al consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Esta propiedad es de solo lectura.

El valor de la propiedad es true si los mensajes se van a entregar al consumidor de mensajes mediante WebSphere MQ Multicast Transport. De lo contrario, el valor es false.

Esta propiedad solo es relevante para una conexión en tiempo real con un intermediario.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Tipo de datos:
System.Boolean

Propiedad de:
MessageConsumer

Indica si los mensajes se entregan al consumidor de mensajes utilizando WebSphere MQ Multicast Transport con una calidad de servicio fiable. Esta propiedad es de solo lectura.

El valor de la propiedad es true si los mensajes se van a entregar al consumidor de mensajes mediante WebSphere MQ Multicast Transport con una calidad de servicio fiable. De lo contrario, el valor es false.

Esta propiedad solo es relevante para una conexión en tiempo real con un intermediario.

XMSC_JMS_MAJOR_VERSION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionMetaData

El número de versión principal de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.

XMSC_JMS_MINOR_VERSION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionMetaData

El número de versión menor de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.

XMSC_JMS_VERSION

Tipo de datos:
Cadena

Propiedad de:
ConnectionMetaData

El identificador de versión de la especificación JMS en la que se basa XMS . Esta propiedad es de solo lectura.

XMSC_MAJOR_VERSION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionMetaData

El número de versión del cliente de XMS . Esta propiedad es de solo lectura.

XMSC_MINOR_VERSION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionMetaData

El número de release del cliente de XMS . Esta propiedad es de solo lectura.

XMSC_PASSWORD

Tipo de datos:
Matriz de bytes

Propiedad de:
ConnectionFactory

Una contraseña que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería. La contraseña se utiliza con la propiedad XMSC_USERID .

De forma predeterminada, la propiedad no está establecida.

Multi Si se está conectando a IBM MQ en Multiplatforms, y establece la propiedad XMSC_USERID de la fábrica de conexiones, debe coincidir con el **`userid`** del usuario conectado. Si no establece estas propiedades, el gestor de colas utiliza el **`userid`** del usuario conectado de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ. Para obtener más información sobre cómo crear una salida de autenticación de cliente, consulte Planificación de la autenticación para una aplicación cliente.

z/OS Para autenticar el usuario cuando se conecta a IBM MQ for z/OS tendrá que utilizar una salida de seguridad.

XMSC_PRIORITY

Tipo de datos:
System.Int32

Propiedad de:
Destino

Nombre utilizado en un URI:
priority

La prioridad de los mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
Un entero dentro del rango de 0, la prioridad más baja, a 9, la prioridad más alta.	Un mensaje enviado al destino tiene la prioridad especificada. La prioridad predeterminada del productor de mensajes, o cualquier prioridad especificada en la llamada Send, se ignora. Si el destino es una cola IBM MQ , el valor del atributo de cola **`DefPriority`** también se ignora.

Valor válido	Significado
XMSC_PRIORITY_AS_APP	Un mensaje enviado al destino tiene la prioridad especificada en la llamada Send. Si la llamada Send no especifica ninguna prioridad, en su lugar, se utiliza la prioridad predeterminada del productor de mensajes. Si el destino es una cola IBM MQ , el valor del atributo de cola **DefPriority** se ignora.
XMSC_PRIORITY_AS_DEST	Si el destino es una cola IBM MQ , un mensaje colocado en la cola tiene la prioridad especificada por el valor del atributo de cola **DefPriority**. La prioridad predeterminada del productor de mensajes, o cualquier prioridad especificada en la llamada Send, se ignora. Si el destino no es una cola IBM MQ , el significado es el mismo que el de XMSC_PRIORITY_AS_APP.

El valor predeterminado es XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport y WebSphere MQ Multicast Transport no realizan ninguna acción basándose en la prioridad de un mensaje.

XMSC_PROVIDER_NAME

Tipo de datos:
Cadena

Propiedad de:
ConnectionMetaData

El proveedor del cliente de XMS . Esta propiedad es de solo lectura.

XMSC_RTT_BROKER_PING_INTERVAL

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El intervalo de tiempo, en milisegundos, transcurrido el cual XMS .NET comprueba la conexión con un servidor de mensajería en tiempo real para detectar cualquier actividad. Si no se detecta ninguna actividad, el cliente inicia una conexión ping; la conexión se cierra si no se detecta ninguna respuesta de ping.

El valor predeterminado de la propiedad es 30000.

XMSC_RTT_CONNECTION_PROTOCOL

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El protocolo de comunicación utilizado para una conexión en tiempo real con un intermediario.

El valor de la propiedad debe ser XMSC_RTT_CP_TCP, lo que significa un conexión en tiempo real con un intermediario sobre TCP/IP. El valor predeterminado es XMSC_RTT_CP_TCP.

XMSC_RTT_HOST_NAME

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de host o la dirección IP del sistema en el que se ejecuta un intermediario.

Esta propiedad se utiliza con la propiedad XMSC_RTT_PORT para identificar el intermediario.

De forma predeterminada, la propiedad no está establecida.

XMSC_RTT_LOCAL_ADDRESS

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para una conexión en tiempo real con un intermediario.

Esta propiedad solo es útil si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y necesita poder especificar qué interfaz se debe utilizar para una conexión en tiempo real. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y la propiedad no está establecida, la interfaz se selecciona de forma aleatoria.

De forma predeterminada, la propiedad no está establecida.

XMSC_RTT_MULTICAST

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory y Destination

Nombre utilizado en un URI:
mulicast

El valor de multidifusión para una fábrica de conexiones o destino. Solo un destino que es un tema puede tener esta propiedad.

Una aplicación utiliza esta propiedad para habilitar la multidifusión con una conexión en tiempo real con un intermediario y, si la multidifusión está habilitada, para especificar la forma precisa en la cual se utiliza la multidifusión para entregar mensajes del intermediario a un consumidor de mensajes. La propiedad no tiene ningún efecto sobre cómo un productor de mensajes envía mensajes al intermediario.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_RTT_MULTICAST_DISABLED	Los mensajes no se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Este valor es el valor predeterminado para un objeto ConnectionFactory.
XMSC_RTT_MULTICAST_ASCF	Los mensajes se entregan a un consumidor de mensajes de acuerdo con el valor de multidifusión para la fábrica de conexiones asociada al consumidor de mensajes. El valor de multidifusión para la fábrica de conexiones se indica en el momento cuando se crea la conexión. Este valor solo es válido para un objeto Destination, y es el valor predeterminado para un objeto Destination.

Valor válido	Significado
XMSC_RTT_MULTICAST_ENABLED	Si el tema se configura para la multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Se utiliza una calidad de servicio fiable si el tema está configurado para la multidifusión fiable.
XMSC_RTT_MULTICAST_RELIABLE	Si el tema se ha configurado para la multidifusión fiable en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport con una calidad de servicio fiable. Si el tema no está configurado para la multidifusión fiable, no puede crear un consumidor de mensajes para el tema.
XMSC_RTT_MULTICAST_NOT_RELIABLE	Si el tema se configura para la multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. No se utiliza una calidad de servicio fiable, aunque el tema se haya configurado para la multidifusión fiable.

XMSC_RTT_PORT

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El número del puerto en el cual un intermediario escucha solicitudes entrantes. En el intermediario, debe configurar el nodo de proceso de mensajes Real-timeInput o Real-timeOptimizedFlow para escuchar en este puerto.

Esta propiedad se utiliza con la propiedad XMSC_RTT_HOST_NAME para identificar el intermediario.

El valor predeterminado de la propiedad es XMSC_RTT_DEFAULT_PORT, o 1506.

XMSC_TIME_TO_LIVE

Tipo de datos:
System.Int32

Propiedad de:
Destino

Nombre utilizado en un URI:

caducidad (para un destino IBM MQ)
timeToLive (para un destino de proveedor de mensajería predeterminado de WebSphere)

El tiempo de vida para los mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
0	Un mensaje enviado al destino nunca caduca.

Valor válido	Significado
Un entero positivo	Un mensaje enviado al destino tiene el tiempo de vida especificado en milisegundos. Se ignora el periodo de vida predeterminado del productor de mensajes, o cualquier periodo de vida especificado en la llamada Send.
XMSC_TIME_TO_LIVE_AS_APP	Un mensaje enviado al destino tiene el periodo de tiempo especificado en la llamada Send. Si la llamada Send no especifica ningún periodo de tiempo, en su lugar, se utiliza el periodo de tiempo predeterminado del productor de mensajes.

El valor predeterminado es XMSC_TIME_TO_LIVE_AS_APP.

XMSC_USERID

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Un identificador de usuario que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería. El identificador de usuario se utiliza con la propiedad XMSC_PASSWORD .

De forma predeterminada, la propiedad no está establecida.

Multi Si está conectándose a IBM MQ for Multiplatforms, y establece la propiedadXMSC_USERID de la fábrica de conexiones, debe coincidir con el **`userid`** del usuario conectado. Si no establece estas propiedades, el gestor de colas utiliza el **`userid`** del usuario conectado de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente que se configura en IBM MQ. Para obtener más información sobre cómo crear una salida de autenticación de cliente, consulte Planificación de la autenticación para una aplicación cliente.

z/OS Para autenticar el usuario cuando se conecta a IBM MQ for z/OS tendrá que utilizar una salida de seguridad.

XMSC_VERSION

Tipo de datos:
Cadena

Propiedad de:
ConnectionMetaData

El identificador de versión de la cliXMSent. Esta propiedad es de sólo lectura.

XMSC_WMQ_BROKER_CONTROLQ

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Nombre de la cola de control utilizada por un intermediario.

El valor predeterminado de la propiedad es SYSTEM.BROKER.CONTROL.QUEUE.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_PUBQ

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la cola supervisada por un intermediario donde las aplicaciones envían mensajes que publican.

El valor predeterminado de la propiedad es SYSTEM.BROKER.DEFAULT.STREAM.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_QMGR

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre del gestor de colas al que está conectado un intermediario.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_SUBQ

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la cola de suscriptor para un consumidor de mensajes no duraderos.

El nombre de la cola de suscriptor debe empezar con los caracteres siguientes:

SYSTEM.JMS.ND.

Si desea que todos los consumidores de mensajes no duraderos compartan una cola de suscriptor, especifique el nombre completo de la cola compartida. Debe existir una cola con el nombre especificado antes de que una aplicación pueda crear un consumidor de mensajes no duraderos.

Si desea que cada consumidor de mensajes no duraderos recupere mensajes de su propia cola de suscriptores exclusiva, especifique un nombre de cola que termine con un asterisco (*). A continuación, cuando una aplicación crea un consumidor de mensajes no duradero, el cliente de XMS crea una cola dinámica para uso exclusivo del consumidor de mensajes. El cliente de XMS utiliza el valor de la propiedad para establecer el contenido del campo **`DynamicQName`** en el descriptor de objetos que se utiliza para crear la cola dinámica.

El valor predeterminado de la propiedad es SYSTEM.JMS.ND.SUBSCRIBER.QUEUE, que significa que XMS utiliza el enfoque de cola compartida de forma predeterminada.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_VERSION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory y Destination

Nombre utilizado en un URI:
brokerVersion

El tipo de intermediario utilizado por la aplicación para una conexión o para el destino. Solo un destino que es un tema puede tener esta propiedad.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_BROKER_V1	La aplicación utiliza un intermediario de publicación/suscripción de IBM MQ . La aplicación también puede utilizar este valor si migra de la publicación/suscripción de IBM MQ a WebSphere Message Broker pero no ha cambiado la aplicación.
XMSC_WMQ_BROKER_V2	La aplicación está utilizando un intermediario de IBM Integration Bus.
XMSC_WMQ_BROKER_UNSPECIFIED	Después de que se haya migrado el intermediario, establezca esta propiedad para que las cabeceras RFH2 dejen de utilizarse. Después de la migración, esta propiedad deja de ser relevante.

El valor predeterminado para una fábrica de conexiones es XMSC_WMQ_BROKER_UNSPECIFIED pero, de forma predeterminada, la propiedad no está establecida para un destino. Establecer la propiedad para un destino sustituirá cualquier valor especificado por la propiedad de la fábrica de conexiones.

XMSC_WMQ_CCDTURL

Tipo de datos:
System.String

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CCDTURL

Nombre abreviado de la herramienta de administración JMS: CCDT

Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de la definición de canal de cliente y, también, especifica cómo se puede acceder al archivo.

De forma predeterminada, esta propiedad no está establecida.

XMSC_WMQ_CCSID

Tipo de datos:
System.Int32

Propiedad de:
Destino

Nombre utilizado en un URI:
CCSID

El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se encuentran las series de datos de caracteres en el cuerpo de un mensaje cuando el cliente XMS reenvía el mensaje al destino. Si se establece para un mensaje individual, la propiedad JMS_IBM_CHARacter_SET altera temporalmente el CCSID especificado para el destino por esta propiedad.

El valor predeterminado de la propiedad es 1208.

Esta propiedad es relevante solo para los mensajes enviados al destino, no para los mensajes recibidos del destino.

XMSC_WMQ_CHANNEL

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CHANNEL

Nombre abreviado de la herramienta de administración JMS: CHAN

El nombre del canal que se va a utilizar para una conexión.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTOPTIONS

Nombre abreviado de la herramienta de administración JMS: CROPT

Esta propiedad especifica las opciones de reconexión de cliente para nuevas conexiones creadas por esta fábrica. Se encuentra en XMSC, y es una de:

- WMQ_CLIENT_RECONNECT_AS_DEF (default). Utilice el valor especificado en el archivo `mqclient.ini`. Establezca el valor utilizando la propiedad **`DefRecon`** dentro de la stanza Channels. Se puede establecer en uno de estos valores:
  1. Sí. Se comporta como la opción WMQ_CLIENT_RECONNECT
  2. NO. Valor por omisión. No especifica ninguna opción de reconexión.
  3. QMGR. Se comporta como la opción WMQ_CLIENT_RECONNECT_Q_MGR
  4. DISABLED. Se comporta como la opción WMQ_CLIENT_RECONNECT_DISABLED
- WMQ_CLIENT_RECONNECT. Reconectarse a cualquiera de los gestores de colas especificados en la lista de nombres de conexión.
- WMQ_CLIENT_RECONNECT_Q_MGR. Se reconecta al mismo gestor de colas al que estaba conectado originalmente. Devuelve `MQRC_RECONNECT_QMID_MISMATCH` si el gestor de colas al que intenta conectarse (especificado en la lista de nombres de conexión) tiene un QMID diferente al del gestor de colas al que estaba conectado originalmente.
- WMQ_CLIENT_RECONNECT_DISABLED. La reconexón está inhabilitada.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTTIMEOUT

Nombre abreviado de la herramienta de administración JMS: CRT

La propiedad XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT sólo es válida para el cliente XMS .NET gestionado.

Esta propiedad especifica la duración de tiempo, en segundos, que una conexión de cliente intenta reconectarse.

Después de intentar reconectarse para esta duración de tiempo, el cliente fallará con MQRC_RECONNECT_FAILED. El valor predeterminado para esta propiedad es XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

El valor predeterminado de esta propiedad es 1800.

XMSC_WMQ_CONNECTION_MODE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

La modalidad a través de la cual una aplicación se conecta a un gestor de colas.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_CM_BINDINGS	Una conexión con un gestor de colas en modalidad de enlaces, para un rendimiento óptimo. Este valor es el valor predeterminado para C/C++.
XMSC_WMQ_CM_CLIENT	Una conexión con un gestor de colas en modalidad cliente, para garantizar una pila totalmente gestionada. Este valor es el valor predeterminado para .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (solo para .NET)	Una conexión con un gestor de colas que fuerza una pila de cliente no gestionada.

XMSC_WMQ_CONNECTION_NAME_LIST

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: CONNECTIONNAMELIST

Nombre abreviado de la herramienta de administración JMS: CNLIST

Esta propiedad especifica los hosts a los que el cliente intenta reconectarse después de que se hayan interrumpido sus conexiones.

La lista de nombres de conexión es una lista separada por comas de pares de host/puerto ip. El valor predeterminado para esta propiedad es WMQ_CONNECTION_NAME_LIST_DEFAULT.

Por ejemplo, `127.0.0.1(1414),host2.example.com(1400)`

El valor predeterminado de esta propiedad es `localhost(1414)`.

XMSC_WMQ_DUR_SUBQ

Tipo de datos:
Cadena

Propiedad de:
Destino

El nombre de la cola de suscriptor para un suscriptor duradero que está recibiendo mensajes del destino. Solo un destino que es un tema puede tener esta propiedad.

El nombre de la cola de suscriptor debe empezar con los caracteres siguientes:

SYSTEM.JMS.D.

Si desea que todos los suscriptores duraderos compartan una cola de suscriptor, especifique el nombre completo de la cola compartida. Una cola con el nombre especificado debe existir antes de que una aplicación pueda crear un suscriptor duradero.

Si desea que cada suscriptor duradero recupere mensajes de su propia cola de suscriptor exclusiva, especifique un nombre de cola que termine con un asterisco (*). A continuación, cuando una aplicación crea un suscriptor duradero, el cliente XMS crea una cola dinámica para uso exclusivo del suscriptor duradero. El cliente de XMS utiliza el valor de la propiedad para establecer el contenido del campo **DynamicQName** en el descriptor de objetos que se utiliza para crear la cola dinámica.

El valor predeterminado de la propiedad es SYSTEM.JMS.D.SUBSCRIBER.QUEUE, que significa que XMS utiliza el enfoque de cola compartida de forma predeterminada.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_ENCODING

Tipo de datos:
System.Int32

Propiedad de:
Destino

Cómo se representan los datos numéricos en el cuerpo de un mensaje cuando el cliente XMS reenvía el mensaje al destino. Si se establece para un mensaje individual, la propiedad JMS_IBM_ENCODING altera temporalmente la codificación especificada para el destino por esta propiedad. La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **Encoding** de un descriptor de mensaje.

Una aplicación puede utilizar las constantes con nombre siguientes para establecer la propiedad:

Constante con nombre	Significado
MQENC_INTEGER_NORMAL	Codificación de entero normal
MQENC_INTEGER_REVERSED	Codificación de entero inverso
MQENC_DECIMAL_NORMAL	Codificación de decimal empaquetado normal
MQENC_DECIMAL_REVERSED	Codificación de decimal empaquetado inverso
MQENC_FLOAT_IEEE_NORMAL	Codificación de coma flotante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codificación de coma flotante IEEE inversa
MQENC_FLOAT_S390	Codificación de coma flotante de arquitectura z/OS
MQENC_NATIVE	Codificación de máquina nativa

Para formar un valor para la propiedad, la aplicación puede añadir tres de estas constantes del modo siguiente:

- Una constante cuyo nombre empieza con MQENC_INTEGER, para especificar la representación de enteros binarios
- Una constante cuyo nombre empieza con MQENC_DECIMAL, para especificar la representación de enteros decimales empaquetados
- Una constante cuyo nombre empieza con MQENC_FLOAT, para especificar la representación de números de coma flotante

De forma alternativa, la aplicación puede establecer la propiedad en MQENC_NATIVE, cuyo valor depende del entorno.

El valor predeterminado de la propiedad es MQENC_NATIVE.

Esta propiedad es relevante solo para los mensajes enviados al destino, no para los mensajes recibidos del destino.

XMSC_WMQ_FAIL_IF_QUIESCE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory y Destination

Nombre utilizado en un URI:
failIfQuiesce

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: FAILIFQUIESCE

Nombre abreviado de la herramienta de administración JMS: FIQ

Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_FIQ_YES	Las llamadas a determinados métodos fallan si el gestor de colas está en un estado de inmovilización. Cuando la aplicación detecta que el gestor de colas está en fase de inmovilización, la aplicación puede completar su tarea inmediata y cerrar la conexión, lo que permite que se detenga el gestor de colas.
XMSC_WMQ_FIQ_NO	No falla ninguna llamada de método porque el gestor está en un estado de inmovilización. Si especifica esta valor, la aplicación no puede detectar que el gestor de colas está en fase de inmovilización. La aplicación podría seguir realizando operaciones en el gestor de colas y, por lo tanto, impedir que se detenga el gestor de colas.

El valor predeterminado para una fábrica de conexiones es XMSC_WMQ_FIQ_YES pero, de forma predeterminada, la propiedad no está establecida para un destino. Establecer la propiedad para un destino sustituirá cualquier valor especificado por la propiedad de la fábrica de conexiones.

XMSC_WMQ_MESSAGE_BODY

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina si una aplicación XMS procesa la MQRFH2 de un mensaje IBM MQ como parte de la carga útil del mensaje (es decir, como parte del cuerpo del mensaje).

Nota: Al enviar mensajes a un destino, la propiedad XMSC_WMQ_MESSAGE_BODY reemplaza a la propiedad de destino XMS existente XMSC_WMQ_TARGET_CLIENT.

Los valores válidos para esta propiedad son:

XMSC_WMQ_MESSAGE_BODY_JMS
Recibir: el tipo de mensaje y el cuerpo de XMS de entrada están determinados por el contenido de la MQRFH2 (si está presente) o la MQMD (si no hay ninguna MQRFH2) en el mensaje IBM MQ recibido.

Enviar: El cuerpo del mensaje XMS de salida contiene una cabecera MQRFH2 prependida y generada automáticamente basándose en las propiedades y los campos de cabecera de XMS Message.

XMSC_WMQ_MESSAGE_BODY_MQ

Recibir: El tipo de mensaje XMS de entrada siempre es ByteMessage, independientemente del contenido del mensaje IBM MQ recibido o del campo de formato del MQMD recibido. El cuerpo del mensaje XMS son los datos de mensaje sin modificar devueltos por la llamada de API del proveedor de mensajería subyacente. El juego de caracteres y la codificación de los datos en el cuerpo del mensaje se determina mediante los campos CodedCharSetId y Encoding del MQMD. El formato de los datos del cuerpo del mensaje se determina mediante el campo Format del MQMD.

Enviar: el cuerpo del mensaje XMS de salida contiene la carga útil de la aplicación tal cual; y no se añade ninguna cabecera IBM MQ generada automáticamente al cuerpo.

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED

Recibir: el cliente XMS determina un valor adecuado para esta propiedad. En la vía de acceso de recepción, este valor es el valor de propiedad WMQ_MESSAGE_BODY_JMS.

Enviar: El cliente XMS determina un valor adecuado para esta propiedad. En la vía de acceso de envío, este valor es el valor de propiedad XMSC_WMQ_TARGET_CLIENT.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Tipo de datos:
System.Int32

Propiedad de:
Destino

Determina qué nivel de contexto de mensaje debe establecer la aplicación XMS . La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.

Los valores válidos para esta propiedad son:

XMSC_WMQ_MDCTX_DEFAULT
Para mensajes de salida, la llamada de la API MQOPEN y la estructura MQPMO no especifican ninguna opción de contexto de mensaje explícita.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_IDENTITY_CONTEXT y la estructura MQPMO especificaMQPMO_SET_IDENTITY_CONTEXT.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_ALL_CONTEXT y la estructura MQPMO especifica MQPMO_SET_ALL_CONTEXT.

De forma predeterminada, esta propiedad está establecida enXMSC_WMQ_MDCTX_DEFAULT.

Nota: Esta propiedad no es relevante cuando una aplicación se conecta a WebSphere Application Server service integration bus.

Las propiedades siguientes requieren que la propiedad XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecida en el valor de propiedad XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT o el valor de propiedad XMSC_WMQ_MDCTX_SET_ALL_CONTEXT al enviar un mensaje para poder tener el efecto deseado:

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

Las propiedades siguientes requieren que la propiedad XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecida en el valor de propiedad XMSC_WMQ_MDCTX_SET_ALL_CONTEXT al enviar un mensaje para poder tener el efecto deseado.

- 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

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina si una aplicación XMS puede extraer o no los valores de los campos MQMD.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_ENABLED_NO
Al enviar mensajes, las propiedades JMS_IBM_MQMD* en un mensaje enviado no se actualizan para reflejar los valores de campo actualizados en el MQMD.

Al recibir mensajes, ninguna de las propiedades JMS_IBM_MQMD* está disponible en un mensaje recibido, aunque el emisor haya establecido algunas o todas las propiedades.

XMSC_WMQ_READ_ENABLED_YES
Al enviar mensajes, todas las propiedades JMS_IBM_MQMD* en un mensaje enviado se actualizan para reflejar los valores de campo actualizados en el MQMD, incluyendo aquellas propiedades que el emisor no ha establecido de forma explícita.

Al recibir mensajes, todas las propiedades JMS_IBM_MQMD* están disponibles en un mensaje recibido, incluyendo aquellas propiedades que el emisor no ha establecido explícitamente.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_READ_ENABLED_NO.

XMSC_WMQ_MQMD_WRITE_ENABLED

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina si una aplicación XMS puede establecer los valores de los campos MQMD o no.

Los valores válidos para esta propiedad son:

XMSC_WMQ_WRITE_ENABLED_NO
Todas las propiedades JMS_IBM_MQMD* se ignoran y sus valores no se copian en la estructura MQMD subyacente.

XMSC_WMQ_WRITE_ENABLED_YES
Se procesan las propiedades JMS_IBM_MQMD*. Sus valores se copian en la estructura MQMD subyacente.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_WRITE_ENABLED_NO.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina si los productores de mensajes están permitidos para utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.

Los valores válidos para esta propiedad son:

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola o tema.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_Q_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_TOPIC_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de tema.

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
Las operaciones de transferencia asíncrona no están permitidas.

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
Las operaciones de transferencia asíncrona están permitidas.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST.

Nota: Esta propiedad no es relevante cuando una aplicación se conecta a WebSphere Application Server service integration bus.

XMSC_WMQ_READ_AHEAD_ALLOWED

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina si los consumidores de mensajes y los navegadores de colas están autorizados para utilizar la lectura anticipada para obtener mensajes no persistentes y no transaccionales de este destino en un almacenamiento intermedio interno antes de recibirlos.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
Determine si se permite la lectura anticipada consultando la definición de cola.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ TOPIC _DEF
Determine si se permite la lectura anticipada consultando la definición de tema.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
Determine si se permite la lectura anticipada consultando la definición de cola o tema.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
La lectura anticipada no está permitida mientras se consumen o navegan mensajes.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
La lectura anticipada está permitida.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Tipo de datos:
System.Int32

Propiedad de:
Destino

Esta propiedad determina, para los mensajes que se están entregando a un escucha de mensajes asíncronos, qué sucede con los mensajes en el almacenamiento de lectura anticipada interno, cuando se cierra el consumidor de mensajes.

Esta propiedad es aplicable en la especificación de opciones de cierre de cola al consumir mensajes de un destino y no es aplicable al enviar mensajes a un destino.

Esta propiedad se ignora para los Examinadores de colas porque, durante la exploración, los mensajes siguen estando disponibles en las colas.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
Solo se completa la invocación del escucha de mensajes actual antes de la devolución, posiblemente dejando mensajes en el almacenamiento intermedio de lectura anticipada interno que, después, se descartan.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
Todos los mensajes del almacenamiento intermedio de lectura anticipada interno se entregan al escucha de mensajes de aplicación antes de la devolución.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ _READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT.

Nota:

Terminación de aplicación anómala
Todos los mensajes del almacenamiento intermedio de lectura anticipada se pierden cuando una aplicación XMS termina de forma abrupta.

Implicaciones para las transacciones
La lectura anticipada está inhabilitada cuando las aplicaciones utilizan transacciones. De esta forma, la aplicación no está viendo ninguna diferencia en el comportamiento cuando utilizando sesiones con transacción.

Implicaciones de las modalidades de conocimiento de sesión
La lectura anticipada está habilitada en una sesión sin transacción cuando las modalidades de acuse de recibo es XMSC_AUTO_ACKNOWLEDGE o bien XMSC_DUPS_OK_ACKNOWLEDGE. La lectura anticipada está inhabilitada si la modalidad de acuse de recibo de sesión es XMSC_CLIENT_ACKNOWLEDGE independientemente de sesiones con transacción o sesiones sin transacción.

Implicaciones para los navegadores de colas y los selectores de navegador de colas
Los navegadores de colas y los selectores de navegador de colas, utilizados en aplicaciones XMS , obtienen la ventaja de rendimiento de la lectura anticipada. El cierre del navegador de colas no degrada el rendimiento porque el mensaje sigue estando disponible en la cola para operaciones posteriores. No hay otras implicaciones para los navegadores de colas y los selectores de navegador de colas aparte de las ventajas de rendimiento de la lectura anticipada.

XMSC_WMQ_HOST_NAME

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: HOSTNAME

Nombre abreviado de la herramienta de administración JMS: HOST

El nombre de host o la dirección IP del sistema en el cual se ejecuta un gestor de colas.

Esta propiedad solo se utiliza cuando una aplicación se conecta a un gestor de colas en modalidad cliente. La propiedad se utiliza con la propiedad XMSC_WMQ_PORT para identificar el gestor de colas.

El valor predeterminado de la propiedad es `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: LOCALADDRESS

Nombre abreviado de la herramienta de administración JMS: LA

Para una conexión a un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se van a utilizar, o ambos.

El valor de la propiedad es una serie con el formato siguiente:

[*nombre_host*][(*puerto_bajo*)[,*puerto_alto*])]

Los significados de las variables son los siguientes:

nombre_host
El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para la conexión.

Proporcionar esta información solo es necesario si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y tendrá que poder especificar qué interfaz se debe utilizar para la conexión. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y no especifica qué interfaz se debe utilizar, la interfaz se selecciona de forma aleatoria.

puerto_bajo
El número del puerto local que se va a utilizar para la conexión.

Si *puerto_alto* también se especifica, *puerto_bajo* se interpreta como el número de puerto inferior en un rango de números de puerto.

puerto_alto
El número de puerto superior en un rango de números de puerto. Se debe utilizar uno de los puertos del rango especificado para la conexión.

La longitud máxima de la serie es 48 caracteres.

Aquí aparecen algunos ejemplos de valores válidos de la propiedad:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_MESSAGE_SELECTION

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Determina si la selección de mensajes la realiza el cliente XMS o el intermediario.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_MSEL_CLIENT	La selección de mensajes ha sido realizada por el cliente de XMS.
XMSC_WMQ_MSEL_BROKER	La selección de mensajes ha sido realizada por el intermediario.

El valor predeterminado es XMSC_WMQ_MSEL_CLIENT.

Esta propiedad solo es relevante en el dominio de publicación/suscripción. La selección de mensajes del intermediario no está soportada si la propiedad XMSC_WMQ_BROKER_VERSION está establecida en XMSC_WMQ_BROKER_V1.

XMSC_WMQ_MSG_BATCH_SIZE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El número máximo de mensajes que se va a recuperar de una cola en un lote cuando se utiliza la entrega de mensajes asíncrona.

Cuando una aplicación está utilizando la entrega de mensajes asíncrona, bajo determinadas condiciones, el cliente XMS recupera un lote de mensajes de una cola antes de enviar cada mensaje individualmente a la aplicación. Esta propiedad especifica el número máximo de mensajes que puede haber en un lote.

El valor de la propiedad es un entero positivo y el valor predeterminado es 10. Considere definir la propiedad en un valor diferente solo si tiene un problema de rendimiento específico que debe abordar.

Si una aplicación está conectada a un gestor de colas sobre una red, aumentar el valor de esta propiedad puede reducir las sobrecargas de la red y los tiempos de respuesta, pero puede aumentar la cantidad de memoria necesaria para almacenar mensajes en el sistema cliente. En cambio, si se reduce el valor de esta propiedad se podría aumentar las sobrecargas de la red y los tiempos de respuesta, pero se podría reducir la cantidad de memoria necesaria para almacenar los mensajes.

XMSC_WMQ_POLLING_INTERVAL

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este valor es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola.

Si con frecuencia sucede esto de que no haya disponible ningún mensaje adecuado para ninguno de los escuchas de mensajes de una sesión, considere aumentar el valor de esta propiedad.

El valor de la propiedad es un entero positivo. El valor predeterminado es 5000.

XMSC_WMQ_PORT

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: PORT

Nombre abreviado de la herramienta de administración JMS: PORT

El número del puerto en el cual un gestor de colas escucha solicitudes entrantes.

Esta propiedad solo se utiliza cuando una aplicación se conecta a un gestor de colas en modalidad cliente. La propiedad se utiliza con la propiedad XMSC_WMQ_HOST_NAME para identificar el gestor de colas.

El valor predeterminado de la propiedad es XMSC_WMQ_DEFAULT_CLIENT_PORT, o 1414.

XMSC_WMQ_PROVIDER_VERSION

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

La versión, release, nivel de modificación y fixpack del gestor de colas al que tiene intención conectarse la aplicación. Los valores válidos para esta propiedad son:

- No especificado

O una serie en uno de los formatos siguientes

- V.R.M.F
- V.R.M
- V.R
- V

donde V, R, M y F son valores enteros mayores que o igual a cero.

Un valor de 7 o superior indica que esta versión está pensada como IBM WebSphere MQ 7.0 ConnectionFactory para conexiones con un gestor de colas de IBM WebSphere MQ 7.0 . Un valor anterior a 7 (por ejemplo "6.0.2.0"), indica que está pensado para ser utilizado con gestores de colas anteriores a la versión 7.0. El valor predeterminado, sin especificar, permite conexiones a cualquier nivel de gestor de colas, determinando las propiedades aplicables y las funciones disponibles basándose en las prestaciones del gestor de colas.

De forma predeterminada, esta propiedad está establecida en "sin especificar".

Nota:

- No se produce ninguna compartición de socket si XMSC_WMQ_PROVIDER_VERSION se establece en 6.2.
- La conexión falla si XMSC_WMQ_PROVIDER_VERSION está establecida en 7 y en el servidor SHARECNV para el canal está establecida en 0.
- Las características específicas de IBM WebSphere MQ 7.0 están inhabilitadas si XMSC_WMQ_PROVIDER_VERSION está establecida en UNSPECIFIED y SHARECNV en 0.

La versión del cliente IBM MQ también desempeña un papel principal con respecto a si una aplicación cliente XMS puede utilizar características específicas de IBM WebSphere MQ 7.0. La tabla siguiente describe el comportamiento.

Nota: Una propiedad de sistema XMSC_WMQ_OVERRIDEPROVIDERVERSION sustituye la propiedad XMSC_WMQ_PROVIDER_VERSION. Esta propiedad se puede utilizar si no puede cambiar el valor de la fábrica de conexiones.

Tabla 880. Cliente XMS - Capacidad para utilizar características específicas de IBM WebSphere MQ 7.0.

#	XMSC_WMQ_PROVIDER_VERSION	Versión del cliente de IBM MQ	Características de IBM WebSphere MQ 7.0
1	no especificada	7	ACTIVADA
2	no especificada	6	DESACTIVADA
3	7	7	ACTIVADA
4	7	6	Excepción
5	6	6	DESACTIVADA
6	6	7	DESACTIVADA

XMSC_WMQ_PUB_ACK_INTERVAL

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Número de mensajes publicados por un publicador antes de que el cliente de XMS solicite un acuse de recibo al intermediario.

Si reduce el valor de esta propiedad, el cliente solicita acuses de recibo con más frecuencia y, por lo tanto, el rendimiento del publicador disminuye. Si se aumenta el valor, el cliente tarda más tiempo en emitir una excepción si el intermediario no se ejecuta correctamente.

El valor de la propiedad es un entero positivo. El valor predeterminado es 25.

XMSC_WMQ_QMGR_CCSID

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El identificador (CCSID) del juego de caracteres codificado, o página de códigos, en el que se intercambian los campos de datos de caracteres definidos en la interfaz de cola de mensajes (MQI) entre el cliente de XMS y el cliente de IBM MQ . Esta propiedad no se aplica a las series de datos de caracteres en los cuerpos de mensajes.

Cuando la aplicación XMS se conecta a un gestor de colas en la modalidad de cliente, el cliente de XMS se enlaza al cliente de IBM MQ. La información intercambiada entre los dos clientes contiene campos de datos de caracteres que están definidos en la MQI. Bajo circunstancias normales, el cliente de IBM MQ da por supuesto que estos campos están en la página de códigos del sistema en el cual se están ejecutando los clientes. Si el cliente de XMS proporciona y espera recibir estos campos en una página de códigos distinta, debe establecer esta propiedad para informar de ello al cliente de IBM MQ.

Cuando el cliente de IBM MQ envía estos campos de datos de caracteres al gestor de colas, los datos incluidos se deben convertir, si es necesario, a la página de códigos utilizada por el gestor de colas. De forma similar, cuando el cliente de IBM MQ recibe estos campos del gestor de colas, los datos aquí incluidos se deben convertir, si es necesario, a la página de códigos en la cual el cliente de XMS espera recibir los datos. El cliente de IBM MQ utiliza esta propiedad para realizar estas conversiones de datos.

De forma predeterminada, la propiedad no está establecida.

Establecer esta propiedad es equivalente a establecer la variable de entorno MQCCSID para un cliente IBM MQ que da soporte a aplicaciones cliente IBM MQ nativas. Para obtener más información sobre esta variable de entorno, consulte MQCCSID.

XMSC_WMQ_QUEUE_MANAGER

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: QMANAGER

Nombre abreviado de la herramienta de administración JMS: QMGR

El nombre del gestor de colas al que conectarse.

De forma predeterminada, la propiedad no está establecida.

XMSC_WMQ_RECEIVE_CCSID

La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora, a menos que XMSC_WMQ_RECEIVE_CONVERSION se establezca en WMQ_RECEIVE_CONVERSION_QMGR.

Tipo de datos:
Entero

Valor:
Cualquier entero positivo.

El valor predeterminado es 1208.

La especificación de un valor `GMO_CONVERT` en un mensaje es opcional. Si se especifica un valor `GMO_CONVERT`, la conversión se realiza de acuerdo con el valor especificado.

XMSC_WMQ_RECEIVE_CONVERSION

La propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.

Tipo de datos:
Entero

Valores:
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (DEFAULT): Realizar conversión de datos sólo en el cliente XMS . La conversión siempre se realiza utilizando la página de códigos 1208.

XMSC_WMQ_RECEIVE_CONVERSION_QMGR: Realizar la conversión de datos en el gestor de colas antes de enviar un mensaje al cliente XMS .

XMSC_WMQ_RECEIVE_EXIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Identifica una salida de recepción de canal que se va a ejecutar.

El valor de la propiedad es una serie que identifica una salida de recepción de canal y tiene el formato siguiente:

`libraryName``(nombrepuntoentrada)`

Donde:

- **`libraryName`** es la vía de acceso completa del archivo `.dll` de salida gestionada

- `nombrepuntoentrada` es el nombre de clase calificado por el espacio de nombres

Por ejemplo: `C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)`

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_RECEIVE_EXIT_INIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Los datos de usuario que se pasan a una salida de recepción de canal cuando se llama.

El valor de la propiedad es una serie. De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad "XMSC_WMQ_RECEIVE_EXIT" en la página 2134 está establecida.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Esta propiedad se utiliza para obtener el nombre del gestor de colas que está conectado.

Cuando se utiliza con una CCDT (tabla de definición de canal de cliente), este nombre podría ser diferente del nombre del gestor de colas especificado en la fábrica de conexiones.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Esta propiedad se llena con el ID del gestor de colas después de la conexión.

XMSC_WMQ_SECURITY_EXIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Identifica una salida de seguridad de canal.

El valor de la propiedad es una serie que identifica una salida de seguridad de canal y tiene el formato siguiente:

`libraryName``(nombrepuntoentrada)`

Donde:

- **`libraryName`** es la vía de acceso completa del archivo .dll de salida gestionada.
- `nombrepuntoentrada` es el nombre de clase calificado por el espacio de nombres

Por ejemplo, C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

La longitud máxima de la serie es 128 caracteres.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_SECURITY_EXIT_INIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Los datos de usuario que se pasan a una salida de seguridad de canal cuando se llama.

La longitud máxima de la serie de datos de usuario es 32 caracteres.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad "XMSC_WMQ_SECURITY_EXIT" en la página 2135 está establecida.

XMSC_WMQ_SEND_EXIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Identifica una salida de emisión de canal.

El valor de la propiedad es una serie. Una salida de emisión de canal tiene el formato siguiente:

`libraryName``(nombrepuntoentrada)`

Donde:

- **`libraryName`** es la vía de acceso completa del archivo .dll de salida gestionada.
- `nombrepuntoentrada` es el nombre de clase calificado por el espacio de nombres

Por ejemplo, `C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)`

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_SEND_EXIT_INIT

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Los datos de usuario que se pasan a las salidas de emisión de canal cuando se llaman.

El valor de la propiedad es una serie de uno o más elementos de datos de usuario separados por comas. De forma predeterminada, la propiedad no está establecida.

Las reglas para especificar datos de usuario que se pasan a una secuencia de salidas de emisión de canal son las mismas que las reglas para especificar datos de usuario que se pasan a una secuencia de salidas de recepción de canal. Por lo tanto, para las reglas consulte "XMSC_WMQ_RECEIVE_EXIT_INIT" en la página 2135.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad "XMSC_WMQ_SEND_EXIT" en la página 2136 está establecida.

XMSC_WMQ_SEND_CHECK_COUNT

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El número de llamadas de envío que se van a permitir entre las comprobaciones de errores de colocación asíncrona dentro de una sesión XMS única sin transacción.

De forma predeterminada, esta propiedad está establecida en 0.

XMSC_WMQ_SHARE_CONV_ALLOWED

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

Objetos aplicables:
Nombre largo de la herramienta de administración JMS: SHARECONVALLOWED

Nombre abreviado de la herramienta de administración JMS: SCALD

Indica si una conexión de cliente puede compartir su socket con otras conexiones XMS de nivel superior del mismo proceso con el mismo gestor de colas, si las definiciones de canal coinciden. Esta propiedad se proporciona para permitir un aislamiento completo de conexiones en sockets separados si es necesario para el desarrollo de aplicaciones, el mantenimiento o por motivos operativos. Si se establece esta propiedad simplemente se indica a XMS para que el socket subyacente se comparta. No indica cuántas conexiones comparten un solo socket. El número de conexiones que comparten un socket se determina mediante el valor SHARECNV que se negocia entre el cliente de IBM MQ y el servidor IBM MQ.

Una aplicación puede establecer las constantes con nombre siguiente para establecer la propiedad:

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - Las conexiones no comparten un socket.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - Las conexiones comparten un socket.

De forma predeterminada, la propiedad está establecida en XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_SSL_CERT_STORES

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Las ubicaciones de los servidores que contienen las listas de revocaciones de certificados (CRL) que se van a utilizar en una conexión SSL a un gestor de colas.

El valor de la propiedad es una lista de uno o más URL separados por comas. Cada URL tiene el formato siguiente:

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Este formato es compatible con, pero con ampliaciones, el formato MQJMS básico.

Es válido tener un `serveraddress`vacío. En este caso, XMS presupone que el valor es la serie "localhost".

Una lista de ejemplos es:

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_CIPHER_SPEC

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la CipherSpec que se va a utilizar en una conexión segura a un gestor de colas.

Las especificaciones de cifrado que puede utilizar con el soporte de IBM MQ TLS se enumeran en la tabla siguiente. Cuando solicite un certificado personal, especifique un tamaño de clave para el par de claves pública y privada. El tamaño de clave que se utiliza durante el reconocimiento SSL es el tamaño almacenado en el certificado, a menos que esté determinado por la CipherSpec, tal como está indicado en la tabla. De forma predeterminada, esta propiedad no está establecida.

Nombre de CipherSpec	Protocolo utilizado	Algoritmo hash	Algoritmo de cifrado	Bits de cifrado	FIPS[1]	Suite B de 128 bits	Suite B de 192 bits
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_CBC_SHA[2]	TLS 1.0	SHA-1	AES	256	Sí	No	No
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	No	No	No
TLS_RSA_WITH_3DES_EDE_CBC_SHA[4]	TLS 1.0	SHA-1	3DES	168	Sí	No	No
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	Sí	No	No
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sí	No	No

Nombre de CipherSpec	Protocolo utilizado	Algoritmo hash	Algoritmo de cifrado	Bits de cifrado	FIPS[1]	Suite B de 128 bits	Suite B de 192 bits
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sí	No	No
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	Sí	No
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	Sí
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No
TLS_RSA_WITH_NULL_NULL	TLS 1.2	Ninguna	Ninguna	0	No	No	No
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No

Notas:

1. Especifica si la CipherSpec es compatible con los estándares federales de procesamiento de información (FIPS) 140-2. Para obtener una explicación de FIPS e información sobre cómo configurar IBM MQ para el funcionamiento compatible con FIPS 140-2, consulte Federal Information Processing Standards (FIPS).
2. Esta CipherSpec no se puede utilizar para proteger una conexión desde IBM MQ Explorer a un gestor de colas a menos que se apliquen los archivos de políticas no restringidas adecuados al JRE utilizado por IBM MQ Explorer.
3. Esta CipherSpec obtuvo el certificado FIPS 140-2 antes del 19 mayo de 2007.
4. Cuando IBM MQ se configura para una operación compatible con FIPS 140-2, se puede utilizar esta CipherSpec para transferir hasta 32 GB de datos antes de que la conexión termine con el error AMQ9288. Para evitar este error, evite utilizar el triple DES (que está en desuso), o habilite el restablecimiento de clave secreta al utilizar esta CipherSpec en una configuración de FIPS 140-2.

Conceptos relacionados

Integridad de datos de mensajes

Tareas relacionadas

Seguridad

Especificación de CipherSpecs

XMSC_WMQ_SSL_CIPHER_SUITE

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la CipherSuite que se va a utilizar en una conexión TLS a un gestor de colas. El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.

Esta propiedad tiene los valores canónicos siguientes:

- 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

Este valor se puede proporcionar como alternativa a XMSC_WMQ_SSL_CIPHER_SPEC.

Si se ha especificado un valor no vacío para XMSC_WMQ_SSL_CIPHER_SPEC, este valor sustituye el valor para XMSC_WMQ_SSL_CIPHER_SUITE. Si XMSC_WMQ_SSL_CIPHER_SPEC no tiene un valor, el valor de XMSC_WMQ_SSL_CIPHER_SUITE se utiliza como la suite de cifrado que se va a proporcionar a GSKit. En este caso, el valor se correlaciona con el valor CipherSpec equivalente, tal como se describe en las correlaciones de nombres CipherSuite y CipherSpec para conexiones XMS con un IBM MQ gestor de colas.

Si tanto XMSC_WMQ_SSL_CIPHER_SPEC como XMSC_WMQ_SSL_CIPHER_SUITE están vacíos, el campo pChDef->SSLCipherSpec se llena con espacios.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_CRYPTO_HW

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Detalles de configuración para el hardware de cifrado conectado al sistema cliente.

Esta propiedad tiene los valores canónicos siguientes:

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

Existe un formato especial para el hardware de cifrado PKCS11 (donde DriverPath, TokenLabel y TokenPassword son series especificadas por usuario):

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS no interpreta ni altera el contenido de la serie. Copia el valor proporcionado, hasta un límite de 256 caracteres de byte único, en el campo MQSCO.CryptoHardware.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_FIPS_REQUIRED

Tipo de datos:
Boolean

Propiedad de:
ConnectionFactory

El valor de esta propiedad determina si una aplicación puede o no puede utilizar suites de cifrado no compatibles con FIPS. Si esta propiedad se establece en true (verdadero), solo se utilizan algoritmos FIPS para la conexión cliente/servidor.

Esta propiedad puede tener los valores siguientes, que se convierten a los dos valores canónicos para MQSCO.FipsRequired:

Tabla 881. Tabla de valores para la propiedad MQSCO.FlipsRequired

Valor	Descripción	Valor correspondiente de MQSCO.FipsRequired
falso	Se puede utilizar cualquier CipherSpec.	MQSSL_FIPS_NO (el valor predeterminado)
true	Solo se pueden utilizar algoritmos criptográficos con certificado FIPS en la CipherSpec que se aplica a esta conexión de cliente.	MQSSL_FIPS_YES

XMS copia el valor relevante en MQSCO.FipsRequired antes de llamar a MQCONNX.

El parámetro MQSCO.FipsRequired sólo está disponible desde IBM WebSphere MQ 6.0. Para IBM WebSphere MQ 5.3, si se establece esta propiedad, XMS no intenta realizar la conexión con el gestor de colas y, en su lugar, emite una excepción adecuada.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_KEY_REPOSITORY

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

La ubicación de un archivo de base de datos de claves en el cual se almacenan claves y certificados.

XMS copia la serie, hasta un límite de 256 caracteres de byte único, en el campo MQSCO.KeyRepository. IBM MQ interpreta esta serie como un nombre de archivo, incluida la vía de acceso completa.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_KEY_RESETCOUNT

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El KeyResetCount representa el número total de bytes sin cifrado enviados y recibidos en una conversación SSL antes de renegociar la clave secreta. El número de bytes incluye la información de control que envía el MCA.

XMS copia el valor que proporcione para esta propiedad en MQSCO.KeyResetCount antes de llamar a MQCONNX.

El parámetro MQSCO.KeyRestCount sólo está disponible en IBM WebSphere MQ 6. Si está ejecutando IBM WebSphere MQ 5.3 y esta propiedad está establecida, XMS no intenta realizar la conexión con el gestor de colas y, en su lugar, emite una excepción adecuada.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

El valor predeterminado de esta propiedad es cero, lo que significa que las claves secretas nunca se renegocian.

Conceptos relacionados

Soporte SSL y TLS para el cliente no gestionado de .NET

Soporte SSL y TLS para el cliente gestionado de .NET

XMSC_WMQ_SSL_PEER_NAME

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de igual que se va a utilizar en una conexión SSL a un gestor de colas.

No hay ninguna lista de valores canónicos para esta propiedad. En su lugar, debe crear esta serie de acuerdo con las reglas para SSLPEER.

Un ejemplo de un nombre de igual es:

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS copia la serie en la página de códigos de byte único y coloca los valores correctos en MQCD.SSLPeerNamePtr y MQCD.SSLPeerNameLength antes de llamar a MQCONNX.

Esta propiedad solo es relevante si la aplicación se conecta a un gestor de colas en modalidad de cliente.

Solo para .NET : Desde IBM MQ 8.0, las conexiones gestionadas con IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas con IBM MQ (WMQ_CM_CLIENT_UNMANAGED) dan soporte a conexiones TLS/SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados
Soporte SSL y TLS para el cliente no gestionado de .NET
Soporte SSL y TLS para el cliente gestionado de .NET
Referencia relacionada
SSLPEERNAME

XMSC_WMQ_SYNCPOINT_ALL_GETS

Tipo de datos:
System.Boolean

Propiedad de:
ConnectionFactory

Indica si se deben recuperar todos los mensajes de colas dentro del control de punto de sincronización.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
falso	Cuando las circunstancias son apropiadas, el cliente de XMS puede recuperar mensajes de colas fuera del control de punto de sincronización.
true	El cliente de XMS debe recuperar todos los mensajes de las colas dentro del control de punto de sincronización.

El valor predeterminado es false.

XMSC_WMQ_TARGET_CLIENT

Tipo de datos:
System.Int32

Propiedad de:
Destino

Nombre utilizado en un URI:
targetClient

Indica si los mensajes enviados al destino contienen una cabecera MQRFH2.

Si una aplicación envía un mensaje que contiene una cabecera MQRFH2, la aplicación receptora debe poder manejar la cabecera.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_TARGET_DEST_JMS	Los mensajes enviados al destino contienen una cabecera MQRFH2. Especifique este valor si la aplicación está enviando los mensajes a otra aplicación XMS , una aplicación IBM MQ classes for JMS o una aplicación IBM MQ nativa que está diseñada para manejar una cabecera MQRFH2 .
XMSC_WMQ_TARGET_DEST_MQ	Los mensajes enviados al destino no contienen una cabecera MQRFH2. Especifique este valor si la aplicación está enviando los mensajes a una aplicación IBM MQ nativa que no se ha diseñado para manejar una cabecera MQRFH2.

El valor predeterminado es XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El prefijo utilizado para formar el nombre de la cola dinámica de IBM MQ que se crea cuando la aplicación crea una cola temporal de XMS .

Las reglas para formar el prefijo son las mismas que las reglas para formar el contenido del campo **DynamicQName** en un descriptor de objeto, pero el último carácter no en blanco debe ser un asterisco (*). Si la propiedad no está establecida, el valor utilizado es `CSQ.*` en z/OS y `AMQ.*` en las otras plataformas. De forma predeterminada, la propiedad no está establecida.

Esta propiedad sólo es relevante en el dominio punto a punto.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory, Destination

Al crear temas temporales, XMS genera una serie de tema con el formato "`TEMP/TEMPTOPICPREFIX/unique_id`", o si esta propiedad contiene el valor predeterminado, se genera esta serie, "`TEMP/unique_id`". Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.

Cualquier serie no nula que conste sólo de caracteres válidos para una serie de tema IBM MQ es un valor válido para esta propiedad.

De forma predeterminada, esta propiedad está establecida en "" (serie vacía).

Nota: Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_TEMPORARY_MODEL

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Nombre de la cola modelo de IBM MQ a partir de la cual se crea una cola dinámica cuando la aplicación crea una cola temporal de XMS .

El valor predeterminado de la propiedad es SYSTEM.DEFAULT.MODEL.QUEUE.

Esta propiedad sólo es relevante en el dominio punto a punto.

XMSC_WMQ_WILDCARD_FORMAT

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory, Destination

Esta propiedad determina qué versión de sintaxis de comodín se va a utilizar.

Cuando se utiliza la publicación/suscripción con IBM MQ '*' y '?' se tratan como comodines. Considerando que '#' y '+' se tratan como comodines cuando se utiliza la publicación/suscripción con IBM Integration Bus. Esta propiedad sustituye la propiedad XMSC_WMQ_BROKER_VERSION.

Los valores válidos para esta propiedad son:

XMSC_WMQ_WILDCARD_TOPIC_ONLY
Reconoce sólo los comodines de nivel de tema, es decir, '#' y '+' se tratan como comodines. Este valor es el mismo que XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY
Reconoce sólo los caracteres comodín, es decir, "*" y '?' se tratan como comodines. Este valor es el mismo que XMSC_WMQ_BROKER_V1.

De forma predeterminada, esta propiedad se establece en XMSC_WMQ_WILDCARD_TOPIC_ONLY.

XMSC_WPM_BUS_NAME

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory y Destination

Nombre utilizado en un URI:
busName

Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.

Para un destino que es un tema, esta propiedad es el nombre del bus de integración de servicios en el cual existe el espacio de tema asociado. Este espacio de tema se especifica mediante la propiedad XMSC_WPM_TOPIC_SPACE.

Si la propiedad no está establecida para un destino, se da por supuesto que la cola o el espacio de tema asociado existe en el bus de integración de servicios al cual se conecta la aplicación.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_CONNECTION_PROTOCOL

Tipo de datos:
System.Int32

Propiedad de:
Conexión

El protocolo de comunicaciones utilizado para la conexión al motor de mensajería. Esta propiedad es de solo lectura.

Los valores posibles de la propiedad son solo siguientes:

Valor	Significado
XMSC_WPM_CP_HTTP	La conexión utiliza HTTP sobre TCP/IP.
XMSC_WPM_CP_TCP	La conexión utiliza TCP/IP.

XMSC_WPM_CONNECTION_PROXIMITY

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El valor de proximidad de conexión para la conexión. Esta propiedad determina cómo de cerca debe estar el motor de mensajería al que se conecta la aplicación en el servidor de programa de arranque.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Valor de proximidad de conexión
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Bus
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Clúster
XMSC_WPM_CONNECTION_PROXIMITY_HOST	Host
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	Servidor

El valor predeterminado es XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Nombre utilizado en un URI:
durableSubscriptionHome

El nombre del motor de mensajería donde se gestionan todas las suscripciones duraderas para una conexión o destino. Los mensajes que se van a entregar a los suscriptores duraderos se almacenan en el punto de publicación del mismo motor de mensajería.

Se debe especificar un inicio de suscripción duradera para una conexión antes de que una aplicación pueda crear una suscriptor duradero que utilice la conexión. Cualquier valor especificado para un destino altera temporalmente el valor especificado para la conexión.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WPM_HOST_NAME

Tipo de datos:
Cadena

Propiedad de:
Conexión

El nombre de host o la dirección IP del sistema que contiene el motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_LOCAL_ADDRESS

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Para una conexión a un bus de integración de servicios, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar, o ambos.

El valor de la propiedad es una serie con el formato siguiente:

[*nombre_host*][(*puerto_bajo*)[,*puerto_alto*])]

Los significados de las variables son los siguientes:

nombre_host
El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para la conexión.

Proporcionar esta información solo es necesario si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y tendrá que poder especificar qué interfaz se debe utilizar para la conexión. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y no especifica qué interfaz se debe utilizar, la interfaz se selecciona de forma aleatoria.

puerto_bajo
El número del puerto local que se va a utilizar para la conexión.

Si *puerto_alto* también se especifica, *puerto_bajo* se interpreta como el número de puerto inferior en un rango de números de puerto.

puerto_alto
El número de puerto superior en un rango de números de puerto. Se debe utilizar uno de los puertos del rango especificado para la conexión.

Aquí aparecen algunos ejemplos de valores válidos de la propiedad:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_ME_NAME

Tipo de datos:
Cadena

Propiedad de:
Conexión

El nombre del motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_NON_PERSISTENT_MAP

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El nivel de fiabilidad de los mensajes no persistente que se envían utilizando la conexión.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Nivel de fiabilidad
XMSC_WPM_MAPPING_AS_DESTINATION	Se determina mediante el nivel de fiabilidad predeterminado especificado para el espacio de cola o tema en el bus de integración de servicios.
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	Mejor esfuerzo no persistente
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	Express no persistente
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	Fiable no persistente
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Fiable persistente
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Seguro persistente

El valor predeterminado es XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

XMSC_WPM_PERSISTENT_MAP

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El nivel de fiabilidad de los mensajes persistentes que se envían mediante la conexión.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Nivel de fiabilidad
XMSC_WPM_MAPPING_AS_DESTINATION	Se determina mediante el nivel de fiabilidad predeterminado especificado para el espacio de cola o tema en el bus de integración de servicios.
XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT	Mejor esfuerzo no persistente
XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT	Express no persistente
XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT	Fiable no persistente
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Fiable persistente
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Seguro persistente

El valor predeterminado es XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

XMSC_WPM_PORT

Tipo de datos:
System.Int32

Propiedad de:
Conexión

El número del puerto en el que estaba a la escucha el motor de mensajería al que está conectada la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_PROVIDER_ENDPOINTS

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Una secuencia de una o más direcciones de punto final de servidores de programa de arranque. Las direcciones de punto final están separadas con comas.

Un servidor de programa de arranque es un servidor de aplicaciones que es responsable de seleccionar el motor de mensajería al que se conecta la aplicación. La dirección de punto final de un servidor de programa de arranque tiene el formato siguiente:

nombre_host:*número_puerto*:*nombre_cadena*

Los significados de los componentes de una dirección de punto final son los siguientes:

nombre_host
El nombre de host o la dirección IP del sistema en el que reside el servidor de programa de arranque. Si no se especifica ningún nombre de host ni ninguna dirección IP, el valor predeterminado es `localhost`.

número_puerto
El número de puerto en el cual escucha el servidor de programa de arranque solicitudes entrantes. Si no se especifica ningún número de puerto, el valor predeterminado es 7276.

nombre_cadena
El nombre de una cadena de transporte del programa de arranque utilizada por el servidor de programa de arranque. Los valores válidos son los siguientes:

Valor válido	Nombre de la cadena de transporte del programa de arranque
XMSC_WPM_BOOTSTRAP_HTTP	BootstrapTunneledMessaging
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	BootstrapSecureMessaging
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

Si no se especifica ningún nombre, el valor predeterminado esXMSC_WPM_BOOTSTRAP_TCP.

Si no se especifica ninguna dirección de punto final, el valor predeterminado es localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_SSL_CIPHER_SUITE

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la CipherSuite que se va a utilizar en una conexión TLS con un motor de mensajería de WebSphere Application Server service integration bus . El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.

Tabla 882. Opciones de CipherSuite para la conexión a un motor de mensajería de WebSphere Application Server service integration bus

Suite de cifrado	Protocolo utilizado
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

Notas:

1. Las suites de cifrado TLS_RSA_WITH_AES_128_CBC_SHA y TLS_RSA_WITH_AES_256_CBC_SHA están soportadas solo en Windows o Solaris. (Esto viene determinado por GSKit.)
2. TLS_RSA_WITH_3DES_EDE_CBC_SHA está en desuso. Sin embargo, se sigue pudiendo utilizar para transferir hasta 32 GB de datos antes de que se termine la conexión con el error AMQ9288. Para evitar este error, tendrá que evitar utilizar el triple DES, o habilitar el restablecimiento de clave secreta cuando se utiliza esta CipherSpec.

No hay ningún valor predeterminado para esta propiedad. Si desea utilizar SSL o TLS, debe especificar un valor para esta propiedad, de lo contrario, la aplicación no puede conectarse correctamente al servidor.

XMSC_WPM_SSL_FIPS_REQUIRED

Tipo de datos:
Boolean

Propiedad de:
ConnectionFactory

El valor de esta propiedad determina si una aplicación puede o no puede utilizar suites de cifrado no compatibles con FIPS. Si esta propiedad está establecida en true, solo se utilizan algoritmos FIPS para la conexión de cliente-servidor.Definir el valor de esta propiedad en TRUE impide que la aplicación utilice suites de cifrado no compatibles con FIPS.

De forma predeterminada, la propiedad está establecida en FALSE (es decir, la modalidad FIPS está desactivada).

XMSC_WPM_SSL_KEY_REPOSITORY

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

Una vía de acceso del archivo que es el archivo de conjunto de claves que contiene las claves públicas o privadas que se van a utilizar en la conexión segura.

Establecer la propiedad del archivo de conjunto de claves en el valor especial de XMSC_WPM_SSL_MS_CERTIFICATE_STORE especifica el uso de la base de datos de claves Microsoft Windows. El uso de la base de datos de claves Microsoft Windows, que se encuentra en **Panel de control** > **Opciones de Internet** > **Contenido** > **Certificados**, elimina la necesidad de una base de datos de archivos de claves separados. El uso de esta constante en Windows x64 y otras plataformas no está permitido.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_LABEL

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El certificado que se va a utilizar al autenticarse con el servidor. Si no se especifica ningún valor, se utiliza el certificado predeterminado.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_PW

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

La contraseña para el archivo de conjunto de claves.

Esta propiedad se puede utilizar como alternativa al uso de XMSC_WPM_SSL_KEYRING_STASH_FILE para configurar la contraseña del archivo de conjunto de claves.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_STASH_FILE

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de un archivo binario que contiene la contraseña del archivo del repositorio de claves.

Esta propiedad se puede utilizar como alternativa al uso de XMSC_WPM_SSL_KEYRING_PW para configurar la contraseña para el archivo de conjunto de claves.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TARGET_GROUP

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de un grupo de destino de motores de mensajería. La naturaleza del grupo de destino viene determinada por la propiedad XMSC_WPM_TARGET_TYPE .

Establezca esta propiedad si desea restringir la búsqueda de un motor de mensajería a un subgrupo de los motores de mensajería en el bus de integración de servicios. Si desea que la aplicación pueda conectarse a cualquier motor de mensajería del bus de integración de servicios, no establezca esta propiedad.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TARGET_SIGNIFICANCE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

La importancia del grupo de destinos de motores de mensajería.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WPM_TARGET_SIGNIFICANCE_ PREFERRED	Se selecciona un motor de mensajería en el grupo de destinos, si hay uno disponible. De lo contrario, se selecciona un motor de mensajería fuera del grupo de destinos, siempre que esté en el mismo bus de integración de servicios.
XMSC_WPM_TARGET_SIGNIFICANCE_ Obligatorio	El motor de mensajería seleccionado debe estar en el grupo de destinos. Si un motor de mensajería del grupo de destino no está disponible, el proceso de conexión falla.

El valor predeterminado de la propiedad es XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El nombre de la cadena de transporte de entrada que debe utilizar la aplicación para conectarse a un motor de mensajería.

El valor de la propiedad puede ser el nombre de cualquier cadena de transporte de entrada que está disponible en el servidor de aplicaciones que aloja el motor de mensajería. La constante con nombre siguiente se proporciona para una de las cadenas de transporte de entrada predefinidas:

Constante con nombre	Nombre de la cadena de transporte
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	InboundBasicMessaging

El valor predeterminado de la propiedad es XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

XMSC_WPM_TARGET_TYPE

Tipo de datos:
System.Int32

Propiedad de:
ConnectionFactory

El tipo del grupo de destinos de motores de mensajería. Esta propiedad determina la naturaleza del grupo de destino identificado por la propiedad XMSC_WPM_TARGET_GROUP .

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WPM_TARGET_TYPE_BUSMEMBER	El nombre del grupo de destinos es el nombre de un miembro de bus. El grupo de destinos está formado por todos los motores de mensajería del miembro de bus.

Valor válido	Significado
XMSC_WPM_TARGET_TYPE_CUSTOM	El nombre del grupo de destinos es el nombre de un grupo definido por usuario de motores de mensajería. El grupo de destinos es para todos los motores de mensajería que están registrados con el grupo definido por usuario.
XMSC_WPM_TARGET_TYPE_ME	El nombre del grupo de destinos es el nombre de un motor de mensajería. El grupo de destinos es el motor de mensajería especificado.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TEMP_Q_PREFIX

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El prefijo utilizado para formar el nombre de la cola temporal que se crea en el bus de integración de servicios cuando la aplicación crea una cola temporal de XMS . El prefijo puede contener hasta 12 caracteres.

El nombre de una cola temporal empieza con los caracteres "_Q" seguidos por el prefijo. El resto del nombre está formado por caracteres generados por el sistema.

De forma predeterminada, la propiedad no está establecida, lo que significa que el nombre de una cola temporal no tiene un prefijo.

Esta propiedad sólo es relevante en el dominio punto a punto.

XMSC_WPM_TEMP_TOPIC_PREFIX

Tipo de datos:
Cadena

Propiedad de:
ConnectionFactory

El prefijo utilizado para formar el nombre de un tema temporal que crea la aplicación. El prefijo puede contener hasta 12 caracteres.

El nombre de un tema temporal empieza con los caracteres "_T" seguidos por el prefijo. El resto del nombre está formado por caracteres generados por el sistema.

De forma predeterminada, la propiedad no está establecida, lo que significa que el nombre de un tema temporal no tiene prefijo.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WPM_TOPIC_SPACE

Tipo de datos:
Cadena

Propiedad de:
Destino

Nombre utilizado en un URI:
topicSpace

El nombre del espacio de tema que contiene el tema. Solo un destino que es un tema puede tener esta propiedad.

De forma predeterminada, la propiedad no está establecida, lo que significa que se presupone el espacio de tema predeterminado.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

Referencia de aplicaciones en desarrollo de Managed File Transfer

Información de referencia para ayudarle a desarrollar aplicaciones para Managed File Transfer.

Ejemplos de uso de fteCreateTransfer para iniciar programas

Puede utilizar el mandato **fteCreateTransfer** para especificar programas a ejecutar antes o después de una transferencia.

Además de utilizar **fteCreateTransfer**, hay otras maneras de invocar un programa antes o después de una transferencia. Para obtener más información, consulte Especificación de programas para ejecutar con MFT.

Todos estos ejemplos utilizan la siguiente sintaxis para especificar un programa:

```
[type:]commandspec[,[retrycount][,[retrywait][,successrc]]]
```

Para obtener más información sobre esta sintaxis, consulte **fteCreateTransfer**: iniciar una nueva transferencia de archivos.

Ejecutar un programa ejecutable

El ejemplo siguiente especifica un programa ejecutable llamado `mycommand` y pasa dos argumentos, a y b, al programa.

```
mycommand(a,b)
```

Para ejecutar este programa en el agente de origen AGENT1 antes de que se inicie la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Ejecutar y reintentar un programa ejecutable

El ejemplo siguiente especifica un programa ejecutable llamado `simple`, que no acepta ningún argumento. Se especifica un valor de 1 para `retrycount` y un valor de 5 para `retrywait`. Estos valores significan que el programa se reintentará una vez si no devuelve un código de retorno satisfactorio, después de una espera de cinco segundos. No se especifica ningún valor para `successrc`, por lo que el único código de retorno satisfactorio es el valor predeterminado de 0.

```
executable:simple,1,5
```

Para ejecutar este programa en el agente de origen AGENT1 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Ejecución de un script Ant y especificación de códigos de retorno satisfactorios

El ejemplo siguiente especifica un script Ant denominado `myscript` y pasa dos propiedades al script. El script se ejecuta utilizando el mandato **`fteAnt`**. El valor de `successrc` se especifica como `>2&<7&!5|0|14`, que especifica que los códigos de retorno de 0, 3, 4, 6 y 14 indican éxito.

```
antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14
```

Para ejecutar este programa en el agente de destino AGENT2 antes de que se inicie la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14"destinationSpecification sourceSpecification
```

Ejecución de un script Ant y especificación de destinos a los que llamar

El ejemplo siguiente especifica un script Ant denominado `script2` y dos destinos, `target1` y `target2`, a los que llamar. También se pasa la propiedad `prop1`, con un valor de `recmfm(F,B)`. Para la coma (,) y los paréntesis en este valor, se debe utilizar la barra inclinada invertida (\) como carácter de escape.

```
antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14
```

Para ejecutar este programa en el agente de destino AGENT2 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14"
destinationSpecification sourceSpecification
```

Utilización de metadatos en un script Ant

Puede especificar una tarea Ant como cualquiera de las siguientes llamadas para una transferencia:

- Origen previo
- Origen posterior
- predestino
- destino posterior

Cuando se ejecuta la tarea Ant , los metadatos de usuario de la transferencia se ponen a disposición utilizando variables de entorno. Puede acceder a estos datos utilizando, por ejemplo, el código siguiente:

```
<property environment="environment" />
<echo>${environment.mymetadata}</echo>
```

donde `mymetadata` es el nombre de algunos metadatos insertados en la transferencia.

Ejecutar un script JCL

El ejemplo siguiente especifica un script JCL llamado ZOSBATCH. Se especifica un valor de 3 para `retrycount`, un valor de 30 para `retrywait` y un valor de 0 para `successrc`. Estos valores significan que el script se reintentará tres veces si no devuelve un código de retorno satisfactorio de 0, con una espera de treinta segundos entre cada intento.

```
jcl:ZOSBATCH,3,30,0
```

Donde ZOSBATCH es un miembro de un PDS denominado MYSYS.JCL, y el archivo `agent.properties` contiene la línea `commandPath=....://'MYSYS.JCL':...`

Para ejecutar este programa en el agente de origen AGENT1 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0
destinationSpecification sourceSpecification
```

Tareas relacionadas

Especificación de programas que se van a ejecutarse con MFT

Referencia relacionada

`fteCreateTransfer`: iniciar una nueva transferencia de archivos

`fteAnt`: ejecutar tareas Ant en MFT

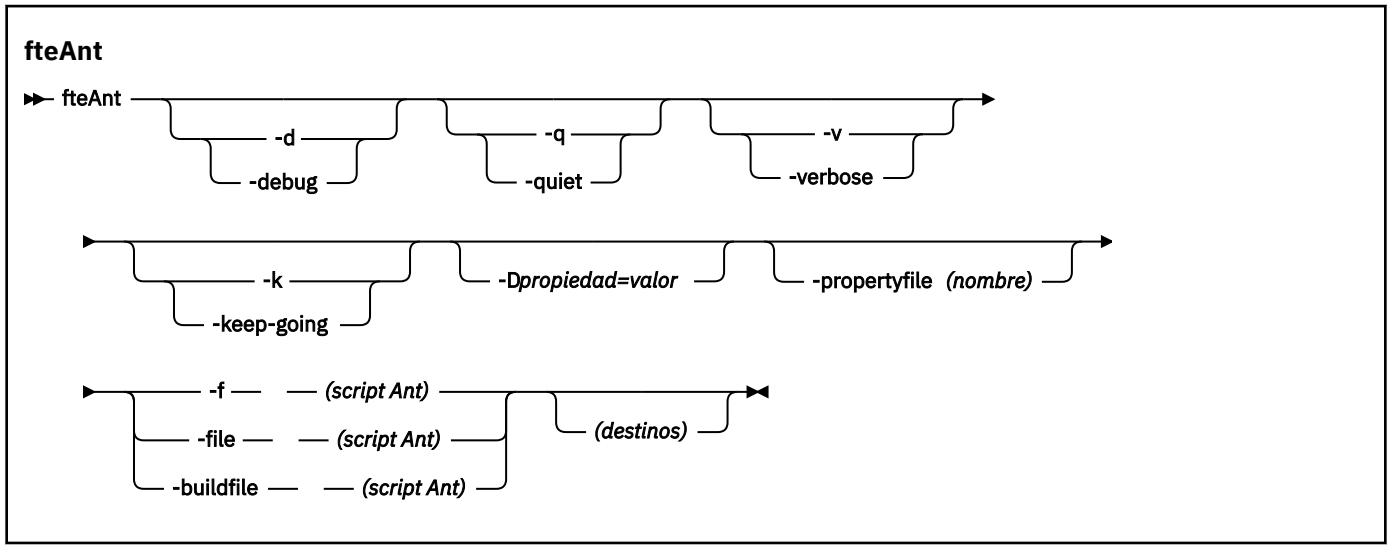
El mandato **`fteAnt`** ejecuta scripts Ant en un entorno que tiene tareas de Managed File Transfer Ant disponibles. A diferencia del mandato **`ant`** estándar, **`fteAnt`** requiere definir un archivo de script.

Tareas y parámetros anidados de MFT Ant

Managed File Transfer proporciona una serie de tareas de Ant que puede utilizar para acceder a las prestaciones de transferencia de archivos. También hay un conjunto de parámetros anidados disponibles; estos parámetros describen conjuntos anidados de elementos que son comunes entre varias de las tareas Ant proporcionadas.

La sintaxis del mandato **`fteAnt`** , los parámetros, el ejemplo de uso y los códigos de retorno se describen en el resto de este tema. Para obtener detalles de las tareas de Ant y los parámetros anidados proporcionados por MFT, consulte los subtemas.

Sintaxis de fteAnt



Parámetros

debug o -d
Opcional. Generar salida de depuración.

-quiet o -q
Opcional. Generar salida mínima.

-verbose o -v
Opcional. Generar salida detallada.

-keep-going o -k
Opcional. Ejecutar todos los destinos que no dependen de destinos anómalos.

-D *propiedad=valor*
Opcional. Utilizar un *valor* para una determinada *propiedad*. Las prioridades establecidas con **-D** tienen prioridad sobre las establecidas en un archivo de propiedades.

Utilice la propiedad **com.ibm.wmqfte.propertyset** para especificar el conjunto de opciones de configuración que se utilizan para las tareas de Ant . Utilice el nombre de un gestor de colas de coordinación no predeterminado como el valor de esta propiedad. A continuación, las tareas de Ant utilizan el conjunto de opciones de configuración que están asociadas con este gestor de colas de coordinación no predeterminado. Si no especifica esta propiedad, se utiliza el conjunto predeterminado de opciones de configuración que se basan en el gestor de colas de coordinación predeterminado. Si especifica el atributo **cmdqm** para una tarea Ant , este atributo tiene prioridad sobre el conjunto de opciones de configuración que se especifican para el mandato **fteAnt** . Este comportamiento se aplica independientemente de si está utilizando el conjunto predeterminado de opciones de configuración o especificando un conjunto con la propiedad **com.ibm.wmqfte.propertyset**.

-propertyfile (*nombre*)
Opcional. Carga todas las propiedades de un archivo con las propiedades **-D** prioritarias.

-f (*script Ant*), -file (*script Ant*), or -buildfile (*script Ant*)
Obligatorio. Especifica el nombre del script Ant que se va a ejecutar.

destinos
Opcional. El nombre de uno o más destinos para ejecutar desde el script Ant. Si no especifica un valor para este parámetro, se ejecuta el destino predeterminado para el script.

-versión
Opcional. Muestra el mandato Managed File Transfer y las versiones de Ant .

-? o -h
Opcional. Muestra la sintaxis del mandato.

Ejemplo

En este ejemplo, se ejecuta el mandato **copy** de destino del script Ant `fte_script.xml` y el mandato graba la salida de depuración en la salida estándar.

```
fteAnt -d -f fte_script.xml copy
```

Códigos de retorno

0
El mandato se ha completado satisfactoriamente.

1
El mandato no ha finalizado correctamente.

También se pueden especificar otros códigos de retorno de estado desde scripts Ant, por ejemplo, utilizando la tarea de error Ant .

Consulte Error para obtener más información.

Tarea fte: awaitoutcome Ant

Espera hasta que se complete una operación **fte:filecopy**, **fte:filemove** o **fte:call**.

Atributos

id
Obligatorio. Identifica la transferencia de la que se espera un resultado. Normalmente, es una propiedad establecida por el atributo idProperty de las tareas fte:filecopy, fte:filemove o fte:call.

rcproperty
Obligatorio. Da nombre a una propiedad en la que se almacena el código de retorno de la tarea**fte:awaitoutcome**.

timeout
Opcional. La cantidad máxima de tiempo, en segundos, de espera hasta que se complete la operación. El tiempo mínimo de espera excedido es un segundo. Si no especifica un valor de tiempo de espera, la tarea **fte:awaitoutcome** espera de forma indefinida hasta que se determine el resultado de la operación.

Ejemplo

En este ejemplo, se inicia una copia de archivo y el identificador se almacena en la propiedad `copy.id`. Mientras la copia está en proceso, puede realizarse otro proceso. La sentencia **fte:awaitoutcome** se utiliza para esperar hasta que se complete la operación de copiar. La sentencia **fte:awaitoutcome** identifica qué operación hay que esperar para utilizar el identificador almacenado en la propiedad `copy.id`. La sentencia **fte:awaitoutcome** almacena un código de retorno indicando el resultado de la operación de copiar en una propiedad denominada `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>
```

Tareas relacionadas
Utilización de Apache Ant con MFT

Tarea fte: call Ant

Puede utilizar la tarea **fte:call** para llamar a scripts y programas de forma autómatica.

Esta tarea le permite enviar una solicitud **fte:call** a un agnete. El agente procesa esta solicitud ejecutando un script o un programa y devolviendo el resultado. Los mandatos para llamar deben ser accesibles para el agente. Asegúrese de que el valor de propiedad commandPath del archivo `agent.properties` incluye la ubicación de los mandatos para llamar. La información de vía de acceso especificada por el elemento anidado del mandato debe ser relativa a las ubicaciones especificadas por la propiedad commandPath. De forma predeterminada, commandPath está vacío, por tanto, el agente no puede llamar a ningún mandato. Para obtener más información sobre esta propiedad, consulte PropiedadcommandPath MFT.

Para obtener más información sobre el archivo `agent.properties` , consulte El archivo MFT `agent.properties`.

Atributos

agent
Obligatorio. Especifica el agente al que enviar la solicitud **fte:call** . Especifique la información de agente con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port*o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties` . Para obtener más información, consulte El archivo MFT command.properties.

Puede utilizar la propiedad **`com.ibm.wmqfte.propertySet`** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte com.ibm.wmqfte.propertySet.

Si no utiliza el atributo cmdqm, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo cmdqm , es decir, `qmgrname@host@port@channel`.

idproperty

Opcional a menos que haya especificado una propiedad `outcome` de `defer`. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore`. Pero debe especificar `idproperty` si también ha especificado una propiedad `outcome` de `defer`.

jobname

Opcional. Asigna un nombre de trabajo a la solicitud **`fte:call`**. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea "Tarea fte: uuid Ant" en la página 2170 para generar nombres de trabajo seudoexclusivos. Si no utiliza el atributo jobname, la tarea utiliza de forma predeterminada el valor de la propiedad `com.ibm.wmqfte.ant.jobName`, si se ha establecido esta propiedad. Si no se establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud **`fte:call`**.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud **`fte:call`**. Si no utiliza el atributo origuser, la tarea adopta de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant.

outcome

Opcional. Determina si la tarea espera a que se complete la operación **`fte:call`** antes de devolver el control al script Ant . Especifique una de las opciones siguientes:

await

La tarea espera a que se complete la operación **`fte:call`** antes de volver. Cuando se especifica una propiedad `outcome` de `await`, el atributo `idproperty` es opcional.

defer

La tarea vuelve tan pronto como se ha enviado la solicitud **`fte:call`** y presupone que el resultado de la operación de llamada se trata más adelante utilizando las tareas awaitoutcome o ignoreoutcome . Cuando se especifica una propiedad `outcome` de `defer`, el atributo `idproperty` es necesario.

ignorar
Si el resultado de la operación **fte:call** no es importante, puede especificar un valor de `ignore`. La tarea volverá en cuanto se haya sometido la solicitud **fte:call**, sin asignar ningún recurso para realizar el seguimiento del resultado del mandato. Cuando se especifica una propiedad `outcome` de `ignore`, el atributo `idproperty` no se puede especificar.

Si no especifica el atributo outcome, la tarea adopta de forma predeterminada el valor `await`.

rcproperty
Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud **fte:call**. El código de resultado refleja el resultado global de la solicitud **fte:call**.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore` o `defer`. Sin embargo, debe especificar `rcproperty` si ha especificado un resultado de `await`.

Parámetros especificados como elementos anidados

fte:command
Especifica el mandato invocado por el agente. Sólo puede asociar un único elemento `fte:command` con una operación **fte:call** determinada. El mandato que se va a invocar debe estar situado en la vía de acceso especificada por la propiedad commandPath en el archivo `agent.properties` del agente.

fte:metadata
Puede especificar metadatos para asociarlos con la operación de llamada. Estos metadatos se graban en los mensajes de registro generados por la operación de llamada. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos.

Ejemplo

Este ejemplo muestra cómo invocar un mandato en AGENT1 que se ejecuta en el gestor de colas QM1. El mandato para invocar es el script `command.sh` y el script se invoca con un único argumento de `xyz`. El mandato `command.sh` está situado en la vía de acceso especificada por la propiedad commandPath en el archivo `agent.properties` del agente.

```
<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>
```

Tareas relacionadas
Utilización de Apache Ant con MFT

fte: cancelar tarea Ant

Cancela una transferencia gestionada o una llamada gestionada de Managed File Transfer. Se puede haber creado una transferencia gestionada mediante las tareas **fte:filecopy** o **fte:filemove**. Se puede haber creado una llamada gestionada mediante la tarea **fte:call**.

Atributos

agent
Obligatorio. Especifica el agente al que se somete la solicitud **fte:cancel**. El valor tiene el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm
Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port*o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties` . Para obtener más información, consulte El archivo MFT command.properties.

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte com.ibm.wmqfte.propertySet.

Si no utiliza el atributo cmdqm, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo cmdqm , es decir, `qmgrname@host@port@channel`.

id
Obligatorio. Especifica el identificador de transferencia de la transferencia que se desea cancelar. Los identificadores de transferencia se generan en el punto en que las tareas fte:filecopy y fte:filemove someten una solicitud de transferencia.

origuser
Opcional. Especifica el identificador del usuario de origen para asociar con la solicitud **cancel**. Si no se utiliza el atributo origuser , la tarea utiliza de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant .

Ejemplo

El ejemplo envía una solicitud **fte:cancel** al gestor de colas de mandatos `qm0`. La solicitud **fte:cancel** tiene como destino `agent1` en el gestor de colas `qm1` para el identificador de transferencia rellenado por la variable `transfer.id`. La solicitud se ejecuta mediante el ID de usuario "bob".

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="agent1@qm1"
            id="${transfer.id}"
            origuser="bob"/>
```

Tareas relacionadas

Utilización de Apache Ant con MFT

Tarea Ant fte:filecopy

La tarea **fte:filecopy** copia archivos entre agentes de Managed File Transfer. El archivo no se suprime del agente de origen.

Atributos

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port*o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties` . Para obtener más información, consulte El archivo MFT command.properties.

Puede utilizar la propiedad **`com.ibm.wmqfte.propertySet`** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte com.ibm.wmqfte.propertySet.

Si no utiliza el atributo cmdqm, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo cmdqm , es decir, `qmgrname@host@port@channel`.

dst

Obligatorio. Especifica el agente de destino para la operación de copiar. Especifique esta información con el formato: `agentname@qmgrname` donde `agentname` es el nombre del agente de destino y `qmgrname` es el nombre del gestor de colas al que está conectado directamente este agente.

idproperty

Opcional a menos que haya especificado una propiedad `outcome` de `defer`. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore`. Pero debe especificar `idproperty` si también ha especificado una propiedad `outcome` de `defer`.

jobname

Opcional. Asigna un nombre de trabajo a la solicitud de copiar. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea "Tarea fte: uuid Ant" en la página 2170 para generar nombres de trabajo seudoexclusivos. Si no utiliza el atributo jobname, la tarea utiliza de forma predeterminada el valor de la propiedad `com.ibm.wmqfte.ant.jobName`, si se ha establecido esta propiedad. Si no establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud de copiar.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud de copiar. Si no utiliza el atributo origuser , la tarea utiliza de forma predeterminada el `ID` de usuario que se utiliza para ejecutar el script Ant .

outcome

Opcional. Determina si la tarea espera a que se complete la operación de copia antes de devolver el control al script Ant . Especifique una de las opciones siguientes:

await

La tarea espera a que se complete la operación de copiar antes de volver. Cuando se especifica una propiedad `outcome` de `await`, el atributo `idproperty` es opcional.

defer
La tarea vuelve tan pronto como se ha enviado la solicitud de copia y presupone que el resultado de la operación de copia se trata más adelante utilizando las tareas “Tarea fte: awaitoutcome Ant” en la página 2157 o “Tarea fte: ignoreoutcome Ant” en la página 2169 . Cuando se especifica una propiedad `outcome` de `defer`, el atributo `idproperty` es necesario.

ignorar
Si el resultado de la operación de copiar no es importante, puede especificar un valor de `ignore`. La tarea volverá en cuanto se haya sometido la solicitud de copiar, sin asignar ningún recurso para realizar el seguimiento del resultado de la transferencia. Cuando se especifica una propiedad `outcome` de `ignore`, el atributo `idproperty` no se puede especificar.

Si no especifica el atributo outcome, la tarea adopta de forma predeterminada el valor `await`.

priority
Opcional. Especifica la prioridad para asociar a la solicitud de copiar. En general, las solicitudes de prioridad más altas tienen prioridad sobre las solicitudes de prioridad más bajas. El valor de prioridad debe estar en el rango entre 0 y 9 (inclusive). Un valor de prioridad de 0 es la prioridad más baja y un valor de 9 es la prioridad más alta. Si no especifica el atributo `priority`, la transferencia adopta una prioridad de 0.

rcproperty
Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud de copiar. El código de resultado refleja el resultado global de la solicitud de copiar.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore` o `defer`. Sin embargo, debe especificar `rcproperty` si especifica un resultado de `await`.

V 9.1.0 **transferRecoveryTimeout**
Opcional. Establece la cantidad de tiempo, en segundos, durante el cual un agente de origen sigue intentando recuperar una transferencia de archivo estancada. Especifique una de las opciones siguientes:

-1
El agente sigue intentando recuperar la transferencia estancada hasta que ésta se lleve a cabo. La utilización de esta opción equivale al comportamiento predeterminado del agente cuando la propiedad no se ha establecido.

0
El agente detiene la transferencia de archivo tan pronto como se inicia la recuperación.

>0
El agente sigue intentando recuperar la transferencia estancada durante el periodo de tiempo en segundos según se haya establecido mediante el valor entero positivo especificado. Por ejemplo,

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filecopy>
```

indica que el agente sigue intentando recuperar la transferencia durante 6 horas desde que se inició la recuperación. El valor máximo para este atributo es 999999999.

Si se especifica el valor de tiempo de espera de recuperación de transferencia de esta forma, se establece en base a la transferencia. Para establecer un valor global para todas las transferencias en una red Managed File Transfer , puede añadir una propiedad a las Propiedades de tiempo de espera de recuperación de transferencia. Para obtener más información, consulte Opción de tiempo de espera para transferencias en recuperación.

src
Obligatorio. Especifica el agente de origen para la operación de copiar. Especifique esta información en el formato: *nombreagente@nombregestorcolas* donde *nombreagente* es el nombre del agente de origen y *nombregestorcolas* es el nombre del gestor de colas con el que está conectado directamente este agente.

Parámetros especificados como elementos anidados

fte:filespec
Obligatorio. Debe especificar al menos una especificación de archivo que identifique los archivos para copiar. Puede especificar que más de una especificación de archivo sea necesaria. Consulte "Elemento anidado fte: filespec Ant" en la página 2171 para obtener más información.

fte:metadata
Puede especificar metadatos para asociarlos con la operación de copiar. Estos metadatos se transportan con la transferencia y se graban en los mensajes de registro generados por la transferencia. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos. Para obtener más información, consulte el tema fte:metadata.

fte:presrc
Especifica una invocación de programa que tendrá lugar en el agente de origen antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:presrc` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:predst
Especifica una invocación de programa que tendrá lugar en el agente de destino antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:predst` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:postsrc
Especifica una invocación de programa que tendrá lugar en el agente de origen después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postsrc` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:postdst
Especifica una invocación de programa que tendrá lugar en el agente de destino después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postdst` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

Si fte:presrc, fte:predst, fte:postsrc, fte:postdst y exits no devuelven un estado de éxito, las reglas se implantarán en el orden especificado:

1. Ejecute las salidas de inicio del origen. Si las salidas de inicio del origen fallan, la transferencia falla y no se ejecuta nada más.
2. Ejecute la llamada previa al origen (si existe). Si la llamada previa al origen falla, la transferencia falla y no se ejecuta nada más.
3. Ejecute las salidas de inicio del destino. Si las salidas de inicio del destino fallan, la transferencia falla y no se ejecuta nada más.
4. Ejecute la llamada previa al destino (si existe). Si la llamada previa al destino falla, la transferencia falla y no se ejecuta nada más.
5. Realice las transferencias de archivos.
6. Ejecute las salidas de finalización del destino. No existe ningún estado de anomalía para estas salidas.
7. Si la transferencia es satisfactoria (si algunos archivos se transfieren satisfactoriamente, se considera satisfactoria), ejecute la llamada posterior al destino (si está presente). Si la llamada posterior al destino falla, la transferencia falla.
8. Ejecute las salidas de finalización del origen. No existe ningún estado de anomalía para estas salidas.
9. Si la transferencia es satisfactoria, ejecute la llamada posterior al origen (si está presente). Si la llamada posterior al origen falla, la transferencia falla.

Ejemplos

En este ejemplo se muestra una transferencia básica de archivos entre `agent1` y `agent2`. El mandato para iniciar la transferencia de archivos se envía a un gestor de colas denominado `qm0`, utilizando una conexión de modalidad de transporte de cliente. El resultado de la operación de transferencia de archivos se asigna a la propiedad denominada `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>
```

En este ejemplo se muestra la misma transferencia de archivos, pero con la incorporación de metadatos y un inicio de programa que tiene lugar en el agente de origen, después de que haya terminado la transferencia.

```
<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>
```

Conceptos relacionados

V 9.1.0 Opción de tiempo de espera para transferencias de archivos en la recuperación

Tareas relacionadas

Utilización de Apache Ant con MFT

Tarea Ant fte:filemove

La tarea **`fte:filemove`** mueve archivos entre agentes de Managed File Transfer. Cuando un archivo se ha transferido satisfactoriamente desde el agente de origen hasta el agente de destino, el archivo de suprime del agente de origen.

Atributos

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port*o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties` . Para obtener más información, consulte El archivo MFT command.properties.

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte com.ibm.wmqfte.propertySet.

Si no utiliza el atributo cmdqm, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo cmdqm , es decir, `qmgrname@host@port@channel`.

dst
Obligatorio. Especifica el agente de destino para la operación de copiar. Especifique esta información con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente de destino y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

idproperty
Opcional a menos que haya especificado una propiedad `outcome` de `defer`. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore`. Pero debe especificar `idproperty` si también ha especificado una propiedad `outcome` de `defer`.

jobname
Opcional. Asigna un nombre de trabajo a la solicitud de mover. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea fte:uuid para generar nombres de trabajo seudoexclusivos. Si no utiliza el atributo jobname, la tarea utiliza de forma predeterminada el valor de la propiedad `com.ibm.wmqfte.ant.jobName`, si se ha establecido esta propiedad. Si no establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud de mover.

origuser
Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud de mover. Si no utiliza el atributo origuser , la tarea utiliza de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant .

outcome
Opcional. Determina si la tarea espera a que se complete la operación de traslado antes de devolver el control al script Ant . Especifique una de las opciones siguientes:

await
La tarea espera a que se complete la operación de mover antes de volver. Cuando se especifica una propiedad `outcome` de `await`, el atributo `idproperty` es opcional.

defer
La tarea vuelve tan pronto como se ha enviado la solicitud de traslado y presupone que el resultado de la operación de traslado se trata más adelante utilizando la tarea “Tarea fte: awaitoutcome Ant” en la página 2157 o “Tarea fte: ignoreoutcome Ant” en la página 2169 . Cuando se especifica una propiedad `outcome` de `defer`, el atributo `idproperty` es necesario.

ignorar
Si el resultado de la operación de mover no es importante, puede especificar un valor de `ignore`. La tarea volverá en cuanto se haya sometido la solicitud de mover, sin asignar ningún recurso para realizar el seguimiento del resultado de la transferencia. Cuando se especifica una propiedad `outcome` de `ignore`, el atributo `idproperty` no se puede especificar.

Si no especifica el atributo outcome, la tarea adopta de forma predeterminada el valor `await`.

priority
Opcional. Especifica la prioridad para asociar a la solicitud de mover. En general, las solicitudes de prioridad más altas tienen prioridad sobre las solicitudes de prioridad más bajas. El valor de prioridad

debe estar en el rango entre 0 y 9 (inclusive). Un valor de prioridad de 0 es la prioridad más baja y un valor de 9 es la prioridad más alta. Si no especifica el atributo `priority`, la transferencia adopta una prioridad de 0.

rcproperty
Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud de mover. El código de resultado refleja el resultado global de la solicitud de mover.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore` o `defer`. Sin embargo, debe especificar `rcproperty` si ha especificado un resultado de `await`.

V 9.1.0 **transferRecoveryTimeout**
Opcional. Establece la cantidad de tiempo, en segundos, durante el cual un agente de origen sigue intentando recuperar una transferencia de archivo estancada. Especifique una de las opciones siguientes:

-1
El agente sigue intentando recuperar la transferencia estancada hasta que ésta se lleve a cabo. La utilización de esta opción equivale al comportamiento predeterminado del agente cuando la propiedad no se ha establecido.

0
El agente detiene la transferencia de archivo tan pronto como se inicia la recuperación.

>0
El agente sigue intentando recuperar la transferencia estancada durante el periodo de tiempo en segundos según se haya establecido mediante el valor entero positivo especificado. Por ejemplo,

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src=agent1@qm1 dst="agent2@qm2"
              rcproperty="move.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filemove
```

indica que el agente sigue intentando recuperar la transferencia durante 6 horas desde que se inició la recuperación. El valor máximo para este atributo es 999999999.

Si se especifica el valor de tiempo de espera de recuperación de transferencia de esta forma, se establece en base a la transferencia. Para establecer un valor global para todas las transferencias en una red Managed File Transfer , puede añadir una propiedad a las Propiedades de tiempo de espera de recuperación de transferencia. Para obtener más información, consulte Opción de tiempo de espera para transferencias en recuperación.

src
Obligatorio. Especifica el agente de origen para la operación de mover. Especifique esta información con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente de origen y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

Parámetros especificados como elementos anidados

fte:filespec
Obligatorio. Debe especificar al menos una especificación de archivo que identifique los archivos para mover. Puede especificar que más de una especificación de archivo sea necesaria. Consulte “Elemento anidado fte: filespec Ant” en la página 2171 para obtener más información.

fte:metadata
Opcional. Puede especificar metadatos para asociarlos con la operación de mover archivo. Estos metadatos se transportan con la transferencia y se graban en los mensajes de registro generados por la transferencia. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos. Para obtener más información, consulte el tema fte:metadata.

fte:presrc
Opcional. Especifica una invocación de programa que tendrá lugar en el agente de origen antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:presrc` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:predst
Opcional. Especifica una invocación de programa que tendrá lugar en el agente de destino antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:predst` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:postsrc
Opcional. Especifica una invocación de programa que tendrá lugar en el agente de origen después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postsrc` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

fte:postdst
Opcional. Especifica una invocación de programa que tendrá lugar en el agente de destino después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postdst` con una transferencia determinada. Consulte el tema invocación de programas para obtener más información.

Si fte:presrc, fte:predst, fte:postsrc, fte:postdst y exits no devuelven un estado de éxito, las reglas se implantarán en el orden especificado:

1. Ejecute las salidas de inicio del origen. Si las salidas de inicio del origen fallan, la transferencia falla y no se ejecuta nada más.
2. Ejecute la llamada previa al origen (si existe). Si la llamada previa al origen falla, la transferencia falla y no se ejecuta nada más.
3. Ejecute las salidas de inicio del destino. Si las salidas de inicio del destino fallan, la transferencia falla y no se ejecuta nada más.
4. Ejecute la llamada previa al destino (si existe). Si la llamada previa al destino falla, la transferencia falla y no se ejecuta nada más.
5. Realice las transferencias de archivos.
6. Ejecute las salidas de finalización del destino. No existe ningún estado de anomalía para estas salidas.
7. Si la transferencia es satisfactoria (si algunos archivos se transfieren satisfactoriamente, la transferencia se considera satisfactoria), ejecute la llamada posterior al destino (si está presente). Si la llamada posterior al destino falla, la transferencia falla.
8. Ejecute las salidas de finalización del origen. No existe ningún estado de anomalía para estas salidas.
9. Si la transferencia es satisfactoria, ejecute la llamada posterior al origen (si está presente). Si la llamada posterior al origen falla, la transferencia falla.

Ejemplos

En este ejemplo se muestra una operación básica de movimiento de archivos entre `agent1` y `agent2`. El mandato para iniciar el movimiento de archivos se envía a un gestor de colas denominado `qm0`, utilizando una conexión de modalidad de transporte. El resultado de la operación de transferencia de archivos se asigna a la propiedad denominada `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>
```

Conceptos relacionados

V 9.1.0 Opción de tiempo de espera para transferencias de archivos en la recuperación

Tareas relacionadas

Utilización de Apache Ant con MFT

Tarea fte: ignoreoutcome Ant

Ignore el resultado de un mandato **fte:filecopy**, **fte:filemove** o **fte:call**. Cuando especifica una tarea **fte:filecopy**, **fte:filemove**o **fte:call** para tener un resultado de defer, la tarea Ant asigna recursos para realizar el seguimiento de este resultado. Si ya no está interesado en el resultado, puede utilizar la tarea **fte:ignoreoutcome** para liberar esos recursos.

Atributos

id

Obligatorio. Identifica el resultado que ha perdido su interés. Normalmente se especifica este identificador utilizando una propiedad que se establece utilizando el atributo idproperty de la tarea "Tarea Ant fte:filecopy" en la página 2161, "Tarea Ant fte:filemove" en la página 2165o "Tarea fte: call Ant" en la página 2158 .

Ejemplo

En este ejemplo se muestra cómo puede utilizar la tarea fte:ignoreoutcome para liberar los recursos asignados al seguimiento del resultado de la tarea "Tarea Ant fte:filecopy" en la página 2161 anterior.

```
<!-- 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}"/>
```

Tareas relacionadas

Utilización de Apache Ant con MFT

Tarea fte: ping Ant

Esta tarea de IBM MQ Managed File Transfer Ant hace ping a un agente para obtener una respuesta y, por lo tanto, determina si el agente puede procesar transferencias.

Atributos

agent

Obligatorio. Especifica el agente al que se somete la solicitud **fte:ping**. El valor tiene el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port*o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo command.properties . Para obtener más información, consulte El archivo MFT command.properties.

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo command.properties se debe utilizar. Para obtener más información, consulte com.ibm.wmqfte.propertySet.

Si no utiliza el atributo cmdqm, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo cmdqm , es decir, `qmgrname@host@port@channel`.

rcproperty
Obligatorio. Nombra una propiedad para almacenar el código de retorno de la tarea **ping**.

timeout
Opcional. El período máximo de tiempo, en segundos, que la tarea espera a que el agente responda. El tiempo de espera mínimo es cero segundos, aunque también se puede especificar un tiempo de espera de menos uno; de este modo, el mandato espera de forma indefinida a que el agente responda. Si no se especifica ningún valor para `timeout`, el valor predeterminado es esperar hasta cinco segundos a que el agente responda.

Ejemplo

Este ejemplo envía una solicitud **fte:ping** al agente `agent1` alojado en `qm1`. La solicitud **fte:ping** espera 15 segundos a que el agente responda. El resultado de la solicitud **fte:ping** se almacena en una propiedad llamada `ping.rc`.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Códigos de retorno

0
El mandato se ha completado satisfactoriamente.

2
Se ha excedido el tiempo de espera del mandato.

Tareas relacionadas
Utilización de Apache Ant con MFT

Tarea fte: uuid Ant

Genera un identificador exclusivo seudoaleatorio y lo asigna a una propiedad determinada. Por ejemplo, puede utilizar este identificador para generar nombres de trabajo para otras operaciones de transferencia de archivos.

Atributos

longitud
Obligatorio. La longitud numérica del UUID que hay que generar. Este valor de longitud no incluye la longitud de cualquier prefijo, que especifica el parámetro **prefix**.

property
Obligatorio. El nombre de la propiedad a la que asignar el UUID generado.

prefix
Opcional. Un prefijo para añadir al UUID generado. Este prefijo no cuenta como parte de la longitud del UUID, tal como lo especifica el parámetro **length**.

Ejemplo

Este ejemplo define un UUID que empieza por las letras ABC seguido de 16 caracteres hexadecimales seudoaleatorios. El UUID se asigna a una propiedad denominada `uuid.property`.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Tareas relacionadas

Utilización de Apache Ant con MFT

Elemento anidado fte: filespec Ant

El parámetro **`fte:filespec`** se utiliza como elemento anidado en otras tareas. Utilice **`fte:filespec`** para describir una correlación entre uno o varios archivos, directorios z/OS o conjuntos de datos de origen y un destino. Normalmente este elemento se utiliza cuando se expresa un conjunto de archivos o directorios z/OS o conjuntos de datos para mover o para copiar.

Anidado por:

- La tarea fte:filecopy
- La tarea fte:filemove

Atributos de especificación de origen

Debe especificar uno de los valores srcfilespec o srcqueue.

srcfilespec
Especifica el origen de la operación de archivo. El valor de este atributo puede incluir un comodín.

srcqueue
Especifica que el origen de la transferencia es una cola. La transferencia mueve datos desde los mensajes almacenados en la cola especificada por este atributo. No puede especificar este atributo si la tarea **`fte:filespec`** está anidada dentro de la tarea **`fte:filecopy`**.

El atributo srcqueue no está soportado cuando el agente de origen es un agente de puente de protocolo.

Atributos de especificación de destino

Debe especificar una de las palabras clave dstdir, dstds, dstfilespace, dstfile, dstqueue o dstpds.

dstdir
Especifica un directorio como destino de una operación de archivo.

z/OS **dstds**
Especifica un conjunto de datos como destino de una operación de archivo.

Este atributo solo está soportado cuando el agente de destino se ejecuta en la plataforma z/OS.

dstfile
Especifica un archivo como destino de una operación de archivo.

dstfilespace
Especifica un espacio de archivos como destino para una operación de archivos.

Este atributo sólo se aplica si el agente de destino es un agente web de IBM MQ 8.0 que tiene acceso al espacio de archivos de la pasarela web.

dstpds

Especifica un conjunto de datos particionados como destino de una operación de archivo.

Este atributo solo está soportado cuando el agente de destino se ejecuta en la plataforma z/OS.

dstqueue

Especifica una cola como destino de una operación de un archivo a un mensaje. Puede incluir, si lo desea, un nombre de gestor de colas en esta especificación, utilizando el formato QUEUE@QUEUEMANAGER. Si no especifica un nombre de gestor de colas, se utiliza el gestor de colas de agente de destino si no ha establecido la propiedad de agente enableClusterQueueInputOutput en true (verdadero). Si la propiedad enableClusterQueueInputOuput se establece en true, el agente de destino utiliza procedimientos de IBM MQ estándares para determinar dónde está ubicada la cola. Debe especificar un nombre de cola válido que exista en el gestor de colas.

Si especifica el atributo dstqueue, no puede especificar los atributos srcqueue porque estos atributos son mutuamente excluyentes.

El atributo dstqueue no está soportado cuando el agente de destino es un agente de puente de protocolo.

Atributos de opciones de origen

srcencoding

Opcional. La codificación del juego de caracteres que el archivo utiliza para realizar transferencias.

Sólo se puede especificar este atributo cuando el atributo conversion tiene un valor de `text`.

Si no se especifica el atributo srcencoding, se utiliza el juego de caracteres del sistema de origen para las transferencias de texto.

srceol

Opcional. El delimitador de fin de la línea utilizado por el archivo que se está transfiriendo. Los valores válidos son los siguientes:

- `CRLF` - Utilizar un carácter de retorno de carro seguido por un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas Windows.
- `LF` - Utilizar un carácter de salto de línea como delimitador de fina de la línea. Este convenio es típico de sistemas UNIX.

Puede especificar este atributo únicamente cuando el atributo conversion se establece en un valor de `text`. Si no se especifica el atributo srceol, las transferencias de texto determinan automáticamente el valor correcto basado en el sistema operativo del agente de origen.

srckeeptrailingspaces

Opcional. Determina si los espacios finales se conservan en los registros de origen leídos desde un conjunto de datos de formato de longitud fija como parte de una transferencia en modalidad de texto. Los valores válidos son los siguientes:

- `true` - los espacios finales se conservan.
- `false` - los espacios finales se eliminan.

Si no especifica el atributo srckeeptrailingspaces, se especifica un valor predeterminado de `false`.

Sólo puede especificar este atributo si también especifica el atributo srcfilespec y establece el atributo conversion en un valor de `text`.

srcmsgdelimbytes

Opcional. Especifica uno o más valores de byte que insertar como delimitador al añadir varios mensajes un archivo de binario. Cada valor se debe especificar como dos dígitos hexadecimales en el rango 00-FF, con el prefijo x. Si se especifican varios bytes, se deben separar con comas. Por ejemplo, `srcmsgdelimbytes="x08,xA4"`. Sólo puede especificar el atributo srcmsgdelimbytes si también

ha especificado el atributo srcqueue. No puede especificar el atributo srcmsgdelimbytes si también ha especificado el valor `text` para el atributo conversion.

srcmsgdelimtext
Opcional. Especifica una secuencia de texto que insertar como delimitador al añadir varios mensajes a un archivo de texto. Se pueden incluir secuencias de escape Java en los literales String del delimitador. Por ejemplo, `srcmsgdelimtext="\u007d\n"`. El delimitador de texto es insertado después de cada mensaje por el agente de origen. El delimitador de texto se cifra en formato binario utilizando el cifrado de origen de la transferencia. Cada mensaje se lee en formato binario, el delimitador cifrado se añade en formato binario al mensaje y el resultado se transfiere en formato binario al agente de destino. Si la página de códigos del agente de origen incluye estados de desplazamiento a teclado estándar y desplazamiento desde teclado estándar, el agente presupone que cada mensaje está en el estado de desplazamiento desde teclado estándar al final del mensaje. En el agente de destino, los datos binarios se convierten de la misma manera que una transferencia de texto de archivo a archivo. También puede especificar el atributo srcmsgdelimtext si ha especificado también el atributo srcqueue y un valor de `text` para el atributo conversion.

srcmsgdelimposition
Opcional. Especifica la posición en la que se inserta el delimitador de texto o binario. Los valores válidos son los siguientes:

- `prefix` - los delimitadores se insertan en el archivo de destino antes de los datos de cada mensaje.
- `postfix` - los delimitadores se insertan en el archivo de destino después de los datos de cada mensaje.

Sólo puede especificar el atributo srcmsgdelimposition si también ha especificado uno de los atributos srcmsgdelimbytes o srcmsgdelimtext.

srcmsggroups
Opcional. Especifica que los mensajes sean agrupados por ID de grupo de IBM MQ. El primer grupo completo se escribe en el archivo de destino. Si no se especifica este atributo, se escribirán todos los mensajes en la cola de origen en el archivo de destino. Sólo puede especificar el atributo srcmsggroups si ha especificado también el atributo srcqueue.

srcqueuetimeout
Opcional. Especifica el tiempo, en segundos, que esperar hasta que se cumpla una de las siguientes condiciones:

- Para que un nuevo mensaje se grabe en la cola.
- Si se ha especificado el atributo srcmsggroups, para que un grupo completo se grabe en la cola.

si no se cumple ninguna de estas condiciones dentro del tiempo especificado por el valor de srcqueuetimeout, el agente de origen deja de leer de la cola y completa la transferencia. Si no se especifica el atributo srcqueuetimeout, el agente de origen deja de leer inmediatamente de la cola de origen si la cola de origen está vacía o, en el caso de haber especificado el atributo srcmsggroups, si no hay ningún grupo completo en la cola. Puede especificar el atributo srcqueuetimeout únicamente si también ha especificado el atributo srcqueue.

Para obtener información sobre cómo establecer el valor srcqueuetimeout, consulte Instrucciones para especificar un tiempo de espera en una transferencia de mensaje a archivo.

z/OS **srcrecdelimbytes**
Opcional. Especifica uno o más valores de byte para insertar como el delimitador al añadir varios registros de un archivo de origen orientado a registros a un archivo binario. Debe especificar cada valor como dos dígitos hexadecimales en el rango 00-FF, con el prefijo x. Si se especifican varios bytes, se deben separar con comas. Por ejemplo:

```
srcrecdelimbytes="x08,xA4"
```

Solo se puede especificar el atributo srcrecdelimbytes si el archivo de origen de la transferencia es un archivo orientado a registro como, por ejemplo, un conjunto de datos de z/OS, y el archivo de destino es un archivo normal no orientado a registro. No puede especificar el atributo srcrecdelimbytes si también ha especificado el valor `text` para el atributo conversion.

srcrecdelimpos
Opcional. Especifica la posición en la que se inserta el delimitador binario. Los valores válidos son los siguientes:

- prefix - los delimitadores se insertan en el archivo de destino antes de los datos de cada registro de archivo de origen orientado a registros.
- postfix - los delimitadores se insertan en el archivo de destino después de los datos de cada registro de archivo de origen orientado a registros.

Puede especificar el atributo srcrecdelimpos sólo si también ha especificado el atributo srcrecdelimbytes.

Atributos de opciones de destino

dstencoding
Opcional. La codificación de juego de caracteres que se utiliza para el archivo transferido.

Sólo se puede especificar este atributo cuando el atributo conversion tiene un valor de `text.`

Si no se especifica el atributo dstencoding, se utiliza el juego de caracteres del sistema de destino para las transferencias de texto.

dsteol
Opcional. El delimitador de fin de la línea que se utiliza para el archivo transferido. Los valores válidos son los siguientes:

- CRLF - Utilizar un carácter de retorno de carro seguido por un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas Windows.
- LF - Utilizar un carácter de salto de línea como delimitador de fina de la línea. Este convenio es típico de sistemas UNIX.

Sólo se puede especificar este atributo cuando el atributo conversion tiene un valor de `text.`

Si no se especifica el atributo dsteol, las transferencias de texto determinan automáticamente el valor correcto basado en el sistema operativo del agente de destino.

dstmsgdelimbytes
Opcional. Especifica el delimitador hexadecimal que utilizar al dividir un archivo binario en varios mensajes. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ LAST_MSG_IN_GROUP definido. El formato para especificar un byte hexadecimal como delimitador es `xNN`, donde N es un carácter en el rango entre 0 y 9 o a y f. Puede especificar una secuencia de bytes hexadecimales como delimitador especificando una lista separada por comas de bytes hexadecimal, por ejemplo: `x3e,x20,x20,xbf`.

Sólo puede especificar el atributo dstmsgdelimbytes si también ha especificado el atributo dstqueue y la transferencia está en modalidad binaria. Sólo puede especificar uno de los atributos dstmsgsize, dstmsgdelimbytes y dstmsgdelimpattern.

dstmsgdelimpattern
Opcional. Especifica la expresión regular Java que se usa al dividir un archivo de texto en varios mensajes. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ LAST_MSG_IN_GROUP definido. El formato para especificar una expresión regular como delimitador es una expresión regular entre paréntesis, `(`*`regular_expression`*`)`, o entre comillas dobles, `"`*`regular_expression`*`"`. Si desea más información, consulte Expresiones regulares utilizadas por MFT.

De forma predeterminada, la longitud de la serie que la expresión regular puede encontrar está limitada por el agente de destino a cinco caracteres. Puede cambiar este comportamiento utilizando la propiedad de agente **maxDelimiterMatchLength**. Para obtener más información, consulte Propiedades avanzadas de agente MFT.

Sólo puede especificar el atributo dstmsgdelimpattern si también ha especificado el atributo dstqueue y la transferencia está en modalidad de texto. Sólo puede especificar uno de los atributos dstmsgsize, dstmsgdelimbytes y dstmsgdelimpattern.

dstmsgdelimposition
Opcional. Especifica la posición en la que está previsto que esté el delimitador de texto o binario. Los valores válidos son los siguientes:

- `prefix` - Los delimitadores se esperan al principio de cada línea.
- `postfix` - Los delimitadores se esperan al final de cada línea.

Sólo puede especificar el atributo dstmsgdelimposition si ha especificado también el atributo dstmsgdelimpattern.

dstmsgincludedelim
Opcional. Especifica si desea incluir en los mensajes el delimitador que se utiliza para dividir el archivo en varios mensajes. Si se especifica el atributo dstmsgincludedelim, el delimitador se incluye al final del mensaje que contiene los datos de archivo que preceden al delimitador. De forma predeterminada, el delimitador no se incluye en los mensajes. Sólo puede especificar el atributo dstmsgincludedelim si también ha especificado uno de los atributos dstmsgdelimpattern y dstmsgdelimbytes.

dstmsgpersist
Opcional. Especifica si los mensajes escritos en la cola de destino son persistentes. Los valores válidos son los siguientes:

- `true` - Escribir mensajes persistentes en la cola de destino. Éste es el valor predeterminado.
- `false` - Escribir mensajes no persistentes en la cola de destino.
- `qdef` - El valor de persistencia se toma del atributo DefPersistence de la cola de destino.

Puede especificar este atributo únicamente cuando también se especifica el atributo dstqueue.

dstmsgprops
Opcional. Especifica si el primer mensaje escrito en la cola de destino tiene la propiedad de mensaje de IBM MQ definida. Los valores posibles son:

- `true` - Definir propiedades de mensajes en el primer mensaje creado por la transferencia.
- `false` - No definir propiedades de mensajes en el primer mensaje creado por la transferencia. Éste es el valor predeterminado.

Para obtener más información, consulte Propiedades de mensajes deMQ establecidas por MFT en mensajes escritos en colas de destino.

Puede especificar este atributo únicamente cuando también se especifica el atributo dstqueue.

dstmsgsize
Opcional. Especifica si dividir el archivo en varios mensajes de longitud fija. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ LAST_MSG_IN_GROUP definido. El tamaño de los mensajes se especifica por el valor de dstmsgsize. El formato de dstmsgsize es *longitudunidades*, where *longitud* es un valor de entero positivo y *unidades* es uno de los valores siguientes:

- B - Bytes. El valor mínimo permitido es dos veces el valor máximo de bytes por carácter de la página de códigos de los mensajes de destino.
- K - Kibibytes. Esto equivale a 1024 bytes.
- M - Mebibytes. El equivalente a 1024 kibibytes.

Si el archivo se transfiere en modalidad de texto, y está en un juego de caracteres de doble byte o un juego de caracteres de varios bytes, el archivo se divide en mensajes en el límite de caracteres más cercano al tamaño de mensaje especificado.

Sólo puede especificar el atributo dstmsgsize si ha especificado también el atributo dstqueue. Sólo puede especificar uno de los atributos dstmsgsize, dstmsgdelimbytes y dstmsgdelimpattern.

dstunsupportedcodepage
Opcional. Especifica la acción que se va a emprender si el gestor de colas de destino, tal como se especifica en el atributo dstqueue, no da soporte a la página de códigos utilizada cuando se transfieren datos de archivos a una cola como una transferencia de texto. Los valores válidos para este atributo son los siguientes:

- `binary` - continuar la transferencia pero no aplicar la conversión de página de códigos a los datos que se están transfiriendo. Especificar este valor equivale a no establecer el atributo de conversión en `text`.
- `fail` - no continuar con la operación de transferencia. El archivo se graba como si no hubiera conseguido transferirse. Éste es el valor predeterminado.

Sólo puede especificar el atributo dstunsupportedcodepage si también ha especificado el atributo dstqueue y un valor de `text` para el atributo conversion.

dsttruncaterecords
Opcional. Especifica que los registros de destino que sean mayores que el atributo de conjunto de datos LRECL se truncan. Si se establece en true, los registros se truncan. Si se establece en false, los registros se ajustan automáticamente. El valor predeterminado es false. Este parámetro sólo es válido para transferencias en modalidad de texto, en las que el destino sea un conjunto de datos.

Otros atributos

checksum
Opcional. Determina el algoritmo que se utiliza para calcular la suma de comprobación de los archivos transferidos.

- `MD5` - utilizar el algoritmo de hash MD5.
- `NONE` - no utilizar un algoritmo de suma de comprobación.

Si no especifica el atributo checksum, se utiliza un valor predeterminado de `MD5`.

conversion
Opcional. Especifica el tipo de conversión que se aplica al archivo cuando se está transfiriendo. Los valores posibles son:

- `binary` - no aplicar ninguna conversión.
- `text` - aplicar la conversión de código de páginas entre los sistemas de origen y destino. Aplicar también la conversión de delimitadores de línea. Los atributos `srcencoding`, `dstencoding`, `srceol` y `dsteol` influyen sobre la conversión que se aplica.

Si no especifica el atributo conversion, se especifica un valor predeterminado de `binary`.

overwrite

Opcional. Determina si la operación puede sobrescribir un z/OS conjunto de datos o un archivo de destino existente. Cuando se especifica un valor de `true`, se sobrescriben los z/OS conjuntos de datos o el archivo de destino existentes. Cuando se especifica un valor de `false`, la existencia de un archivo z/OS o un conjunto de datos duplicado en los resultados en el destino hace que la operación falle. Si no se especifica el atributo overwrite, se especifica un valor predeterminado de `false`.

recurse
Opcional. Determina si la transferencia de archivos se repite en los subdirectorios. Cuando se especifica un valor de `true`, la transferencia se repite en los subdirectorios. Cuando se especifica un valor de `false`, la transferencia no se repite en los subdirectorios. Si no se especifica el atributo recurse, se especifica un valor predeterminado de `false`.

Ejemplo

Este ejemplo especifica un fte: filespec con un archivo de origen de `file1.bin` y un archivo de destino de `file2.bin.`

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

Tareas relacionadas

Utilización de Apache Ant con MFT

fte: metadata Ant elementos anidados

Los metadatos se utilizan para transmitir información adicional definida por el usuario con una operación de transferencia de archivo.

Consulte "Metadatos para salidas de usuario de MFT" en la página 2181 para obtener más información sobre cómo Managed File Transfer utiliza los metadatos.

Anidado por:

- La tarea fte:filecopy
- La tarea fte:filemove
- La tarea fte:call

Parámetros especificados como elementos anidados

fte:entry
Debe especificar al menos una entrada dentro del elemento anidado fte:metadata . Puede optar por especificar más de una entrada. Las entradas asocian un nombre de clave a un valor. Las claves deben ser exclusivas en un bloque de fte:metadata

Atributos de entrada

name
Obligatorio. El nombre de la clave que pertenece a esta entrada. Este nombre debe ser exclusivo para todos los parámetros anidados **`entry`** dentro de un elemento fte:metadata.

value
Obligatorio. El valor que se asigna a esta entrada.

Ejemplo

Este ejemplo muestra una definición fte:metadata que contiene dos entradas.

```
<fte:metadata>
    <fte:entry name="org.foo.partColor" value="red"/>
    <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Tareas relacionadas

Utilización de Apache Ant con MFT

Elementos anidados de invocación de programa

Se pueden iniciar programas utilizando uno de cinco elementos anidados: `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc` y `fte:command`. Estos elementos anidados ordenan a un agente que llame a un programa externo como parte del proceso. Para poder iniciar un programa, antes debe asegurarse de que el mandato está en una ubicación especificada por la propiedad commandPath en el archivo `agent.properties` del agente que ejecute el mandato.

Aunque cada elemento de invocación de programa tiene un nombre diferente, comparten el mismo conjunto de atributos y el mismo conjunto de elementos anidados. Los programas se pueden iniciar mediante las tareas Ant **`fte:filecopy`**, **`fte:filemove`**y **`fte:command`** .

No puede invocar programas desde un agente de puente Connect:Direct.

Tareas Ant que pueden invocar programas:

- La tarea fte:filecopy anida parámetros de invocación de programa mediante los elementos anidados `fte:predst`, `fte:postdst`, `fte:presrc` y `fte:postsrc`.
- La tarea fte:filemove anida parámetros de invocación de programa mediante los elementos anidados `fte:predst`, `fte:postdst`, `fte:presrc` y `fte:postsrc`.
- La tarea fte:call anida parámetros de invocación de programa mediante el elemento anidado `fte:command`.

Atributos

mandato

Obligatorio. Nombra el programa que se invoca. Para que el agente pueda ejecutar un mandato, el mandato debe encontrarse en una ubicación especificada por la propiedad commandPath en el archivo `agent.properties` del agente. Para obtener más información, consulte commandPath MFT property. La información sobre la vía de acceso especificada en el atributo `command` se considera relativa a una ubicación especificada por la propiedad commandPath. Cuando `type` es `executable`, se espera un programa ejecutable; en caso contrario, se espera un script apropiado al tipo de llamada.

retrycount

Opcional. El número de veces que se reintenta llamar al programa si el programa no devuelve un código de retorno satisfactorio. El programa especificado por el atributo `command` se invoca como máximo este número de veces. El valor asignado a este atributo debe ser un valor no negativo. Si no especifica el atributo `retrycount`, se utiliza un valor predeterminado de cero.

retrywait

Opcional. El tiempo de espera, en segundos, antes de intentar de nuevo la invocación de programa. Si el programa especificado por el atributo `command` no devuelve un código de retorno satisfactorio y el atributo `retrycount` especifica un valor distinto de cero, este parámetro determina el tiempo de espera entre reintentos. El valor asignado a este atributo debe ser un valor no negativo. Si no especifica el atributo `retrywait`, se utiliza un valor predeterminado de cero.

successrc

Opcional. El valor de este atributo se utiliza para determinar cuándo la invocación de programa se ejecuta satisfactoriamente. El código de retorno del proceso para el mandato se evalúa mediante esta expresión. El valor puede estar compuesto de una o más expresiones combinadas con un carácter de barra vertical (|) para indicar booleano OR, o un ampersand (&) para indicar el valor booleano AND. Cada expresión puede ser uno de los tipos siguientes de expresión:

- Un número que indica una prueba de igualdad entre el código de retorno del proceso y el número.
- Un número con el prefijo ">" para indicar una prueba mayor que entre el número y el código de retorno del proceso.
- Un número con el prefijo de un carácter "<" para indicar una prueba menor que entre el número y el código de retorno del proceso.

- Un número que tiene como prefijo un carácter "!" para indicar una prueba de 'no igual a' entre el número y el código de retorno de proceso.

Por ejemplo: `>2&<7&!5|0|14` se interpreta como los siguientes códigos de retorno satisfactorios: 0, 3, 4, 6, 14. Los demás códigos de retorno se interpretan como no satisfactorios. Si no especifica el atributo `successrc`, se utiliza un valor predeterminado de cero. Esto significa que se interpreta que el mandato se ha ejecutado satisfactoriamente si, y solamente si, dicho mandato devuelve un código de cero.

tipo

Opcional. El valor de este atributo especifica qué tipo de programa se está invocando. Especifique una de las opciones siguientes:

executable

La tarea invoca un programa ejecutable. Puede tener argumentos adicionales especificados mediante el elemento anidado arg. Se espera que el programa sea accesible en commandPath y, en caso necesario, que tenga establecido el permiso de ejecución. Se puede llamar a los scripts UNIX siempre que especifiquen un programa de shell (por ejemplo, la primera línea del archivo de script de shell es: `#!/bin/sh`). La salida del mandato escrita en stderr o stdout se envía al registro de Managed File Transfer para la llamada. No obstante, la cantidad de salida de datos está limitada por la configuración del agente. El valor predeterminado es 10K bytes de datos, pero puede alterar temporalmente este valor predeterminado utilizando la propiedad de agente: maxCommandOutput.

antscript

La tarea ejecuta el script Ant especificado, utilizando el mandato **`fteAnt`** . Se pueden especificar propiedades mediante el elemento anidado property. Los destinos Ant se pueden especificar utilizando el elemento anidado target . Se espera que el script Ant sea accesible en commandPath. La salida de Ant escrita en stderr o stdout se envía al registro de Managed File Transfer para la llamada. No obstante, la cantidad de salida de datos está limitada por la configuración del agente. El valor predeterminado es 10K bytes de datos, pero puede alterar temporalmente este valor mediante la propiedad de agente: maxCommandOutput.

z/OS **jcl**

El valor `jcl` solo está soportado en z/OS y ejecuta el script JCL de z/OS especificado. El JCL se somete como un trabajo y requiere que haya una tarjeta de trabajo. Cuando el trabajo se somete satisfactoriamente, la salida del mandato JCL, grabada en el registro de Managed File Transfer, contiene el siguiente texto: JOB *nombre_trabajo*(*id_trabajo*), donde:

- *nombre_trabajo* es el nombre del trabajo identificado por la tarjeta de trabajo del JCL.
- *id_trabajo* es el identificador de trabajo generado por el sistema z/OS.

Si el trabajo no se puede someter satisfactoriamente, el mandato de script JCL no se ejecuta correctamente y graba un mensaje en el registro indicando la razón del error (por ejemplo, no hay ninguna tarjeta de trabajo). Para saber si el trabajo se ha ejecutado o se ha completado satisfactoriamente, utilice un servicio del sistema, como, por ejemplo, SDSF. Managed File Transfer no proporciona esta información porque sólo somete el trabajo; a continuación, el sistema determina cuándo debe ejecutarse el trabajo y cómo se presenta la salida del trabajo. Puesto que un script JCL se somete como un trabajo por lotes, no es aconsejable especificar `jcl` para un elemento anidado `presrc` o `predst`, porque lo único que sabe es que el trabajo se ha sometido satisfactoriamente pero no si se ha ejecutado satisfactoriamente hasta el final antes de que se inicie la transferencia. No hay ningún elemento anidado que sea válido con un tipo de `jcl`.

El siguiente ejemplo muestra un trabajo de 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
```

Parámetros especificados como elementos anidados

fte:arg
Sólo es válido cuando el valor del atributo `type` es executable. Utilice elementos `fte:arg` anidados para especificar argumentos para el programa que se está invocando como parte de la invocación de programa. Los argumentos del programa se crean a partir de los valores especificados por los elementos `fte:arg` en el orden en que se encuentran los elementos `fte:arg`. Si lo desea, puede especificar cero o más elementos `fte:arg` como elementos anidados de una invocación de programa.

fte:property
Sólo es válido cuando el valor del atributo `type` es antscript. Utilice los atributos `name` y `value` de los elementos `fte:property` anidados para pasar pares nombre-valor al script Ant . Si lo desea, puede especificar cero o más elementos `fte:property` como elementos anidados de una invocación de programa.

fte:target
Sólo es válido cuando el valor del atributo `type` es antscript. Especifique un destino en el script Ant al que llamar. Si lo desea, puede especificar cero o más elementos `fte:target` como elementos anidados de una invocación de programa.

Atributos de Arg

valor
Obligatorio. El valor del argumento que se pasa al programa que se está invocando.

Atributos de Property

nombre
Obligatorio. El nombre de una propiedad que se debe pasar al script Ant .

valor
Obligatorio. El valor que se debe asociar con el nombre de propiedad que se pasa al script Ant .

Ejemplos

Este ejemplo muestra una invocación de programa `fte:postsrc` que se especifica como parte de una tarea fte:filecopy. La invocación de programa es para un programa denominado `post.sh` y se le suministra un único argumento de `/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>
```

Este ejemplo muestra una invocación de programa `fte:command` que se especifica como parte de una tarea fte: call . La invocación de programa es para un archivo ejecutable llamado `command.sh`, al que no se pasa ningún argumento de línea de mandatos. Si `command.sh` no devuelve un código de retorno satisfactorio de 1, se intenta ejecutar de nuevo el mandato al cabo de 30 segundos.

```
<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>
```

Este ejemplo muestra una invocación de programa `fte:command` que se especifica como parte de una tarea fte: call . La invocación de programa es para los destinos de copia y compresión en un script Ant denominado `script.xml`, al que se le pasan dos propiedades.

```
<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>
```

Tareas relacionadas

Especificación de programas que se van a ejecutarse con MFT

Utilización de Apache Ant con MFT

Salidas de usuario de MFT para referencia de personalización

Información de referencia para ayudarle a configurar salidas de usuario para Managed File Transfer.

Conceptos relacionados

Salidas de usuario de origen y destino de MFT

Metadatos para salidas de usuario de MFT

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

Metadatos del entorno

Los metadatos del entorno se transfieren a todas las rutinas de salida de usuario y describen el entorno de ejecución del agente desde el que se invoca la rutina de salida de usuario. Estos metadatos son de sólo lectura y no se pueden actualizar mediante ninguna rutina de salida de usuario.

Tabla 883. Metadatos del entorno

Clave	Descripción
AGENT_CONFIGURATION_DIRECTORY_KEY	El nombre del directorio que contiene la información de configuración del agente.
AGENT_PRODUCT_DIRECTORY_KEY	El nombre del directorio en el que se ha instalado el código del agente.
AGENT_VERSION_KEY	Número de versión para el tiempo de ejecución del agente que invoca la rutina de salida.

Los nombres de clave y valor suministrados en la Tabla 1 son constantes que se definen en la interfaz EnvironmentMetaDataConstants.

Metadatos de transferencia

Los metadatos de transferencia se transfieren a todas las rutinas de salida de usuario. Los metadatos constan de valores suministrados por el sistema y valores suministrados por el usuario. Si cambia cualquier valor proporcionado por el sistema, estos cambios serán ignorados. Los valores proporcionados por el usuario iniciales para la salida de usuario de inicio de transferencia de origen se basan en

los valores proporcionados al definir la transferencia. El agente de origen puede cambiar los valores suministrados por el usuario como parte del proceso de la salida de usuario de inicio de la transferencia de origen. Esta rutina de salida se invoca antes de que se inicie toda la transferencia de archivos. Estos cambios se utilizan en llamadas posteriores a otras rutinas de salida que están relacionadas con esa transferencia. Los metadatos se transferencia se aplican a una transferencia entera.

Aunque las salidas de usuario pueden leer valores de los metadatos de transferencia, únicamente la salida de usuario de inicio de la transferencia de origen puede cambiar los metadatos de transferencia.

No puede utilizar metadatos de transferencia para difundir información entre transferencias de archivos diferentes.

Los metadatos de transferencia proporcionados por el sistema se detallan en la Tabla 2:

Tabla 884. Metadatos de transferencia

Clave	Descripción
DESTINATION_AGENT_KEY	El nombre del agente que es el destino de la transferencia.
JOB_NAME_KEY	El nombre de trabajo asociado a la solicitud de transferencia
MQMD_USER_KEY	El campo de usuario MQMD del mensaje empleado para someter la solicitud de transferencia
ORIGINATING_HOST_KEY	El nombre de host especificado como el nombre de host de origen en la solicitud de transferencia
ORIGINATING_USER_KEY	El nombre de usuario especificado como el ID de usuario de origen en la solicitud de transferencia
SOURCE_AGENT_KEY	El nombre del agente que es el origen de la transferencia
TRANSFER_ID_KEY	El identificador de la transferencia

Los nombres de clave y valor suministrados en la Tabla 2 son constantes que se definen en la interfaz TransferMetaDataConstants.

Metadatos de archivo

Los metadatos de archivo se transfieren a la salida del inicio de la transferencia de origen como parte de la especificación de archivo. Existen metadatos de archivo separados para los archivos de origen y destino.

No puede utilizar metadatos para difundir información entre transferencias de archivos diferentes.

Tabla 885. Metadatos de archivo

Clave	Valores permitidos	Descripción
CONVERT_LINE_SEPARATORS		El valor de clave utilizado para las transferencias de texto para indicar si las secuencias de separador de línea CRLF (retorno de carro, salto de línea) o LF (salto de línea) en los datos de origen se convierten a la secuencia de separador de línea de separación en el destino.
DELIMITER_KEY		El valor de clave utilizado para definir un delimitador para separar datos de registro cuando se transfieren datos orientado a registros a archivos normales. También se utiliza para transferencias de mensaje a archivo y de archivo a mensaje.

Tabla 885. Metadatos de archivo (continuación)		
Clave	**Valores permitidos**	**Descripción**
DELIMITER_POSITION_KEY	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	Utilizar con DELIMITER_KEY para definir la posición del delimitador; prefijo o postfijo.
DELIMITER_TYPE_KEY	DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE DELIMITER_TYPE_SIZE_VALUE	Utilice esta opción con DELIMITER_KEY para definir el tipo de delimitador.
DESTINATION_EXIST_KEY	DESTINATION_EXIST_KEY_ERROR_VALUE DESTINATION_EXIST_KEY_OVERWRITE_VALUE	Determina el comportamiento de la transferencia de archivos si el archivo de destino existe.
FILE_ALIAS_KEY		Valor de clave utilizado para definir un alias para el archivo que se está transfiriendo.
FILE_CHECKSUM_METHOD_KEY	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Determina el método de suma de comprobación que se utiliza cuando se transfiere el archivo.
FILE_CONVERSION_KEY	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	Determina el tipo de conversión que se aplica al contenido del archivo.
FILE_ENCODING_KEY		Determina la codificación que se utiliza para un archivo de texto.
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	Determina la secuencia de caracteres que indica el final de una línea: < LF > o < CR> < LF>.
FILE_SPACE_ALIAS		Determina el alias de un archivo en el espacio de archivos. **Nota:** Estos metadatos pueden utilizarse sólo si FILE_TYPE_KEY es FILE_TYPE_FILE_SPACE_VALUE
FILE_SPACE_NAME		Determina el nombre del espacio de archivos. **Nota:** Estos metadatos pueden utilizarse sólo si FILE_TYPE_KEY es FILE_TYPE_FILE_SPACE_VALUE
FILE_TYPE_KEY	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Determina la especificación del archivo de destino, cola o espacio de archivos.
GROUP_ID_KEY		Valor de clave utilizado para transferencias de mensaje a archivo para determinar el grupo de mensajes a leer de la cola de origen. Este atributo sólo es válido cuando el valor de USE_GROUPS_KEY es USE_GROUPS_TRUE_VALUE.
INCLUDE_DELIMITER_IN_MESSAGE_KEY	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si se deben incluir los delimitadores que se han utilizado para dividir el archivo en varios mensajes al final de los mensajes. Este atributo sólo es válido cuando el valor de DELIMITER_TYPE_KEY es DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE.

Tabla 885. Metadatos de archivo (continuación)		
Clave	**Valores permitidos**	**Descripción**
INSERT_RECORD_LINE_SEPARATOR_KEY		Valor de clave utilizado para las transferencias de texto desde archivos orientados a registros para especificar si se insertan separadores de línea en los datos después de cada registro.
KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	Valor de clave utilizado para determinar si los espacios de cola se eliminan de los registros leídos en los conjuntos de datos de formato de longitud fija.
NEW_RECORD_ON_LINE_SEPARATOR_KEY		Valor de clave utilizado para transferencias de texto a archivos orientados a registros para especificar si los separadores de línea en los datos se incluyen en los datos de registro o producen un nuevo registro (y no se graban).
PERSISTENT_KEY	PERSISTENT_TRUE_VALUE PERSISTENT_FALSE_VALUE PERSISTENT_QDEF_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si los mensajes son persistentes.
SET_MQ_PROPS_KEY	SET_MQ_PROPS_TRUE_VALUE SET_MQ_PROPS_FALSE_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si las propiedades de mensaje IBM MQ se establecen en el primer mensaje de un archivo y los mensajes se graban en la cola cuando se produce un error.
UNRECOGNISED_CODE_PAGE_KEY	UNRECOGNISED_CODE_PAGE_FAIL_VALUE UNRECOGNISED_CODE_PAGE_BINARY_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si falla una transferencia de modalidad de texto o se realiza una conversión, en el caso de que el gestor de colas de destino no reconozca la página de códigos de los datos.
USE_GROUPS_KEY	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	Valor de clave utilizado para transferencias de mensaje a archivo para determinar si sólo se debe transferir un grupo completo de mensaje de la cola de origen.
WAIT_TIME_KEY		Valor de clave utilizado para transferencias de mensaje a archivo para determinar el tiempo, en segundos, que el agente de origen debe esperar a que se produzca uno de los casos siguientes: • Aparezca un mensaje en la cola de origen, si la cola está vacía o se ha quedado vacía, cuando el valor de USE_GROUPS_KEY es FALSE. • Aparezca un grupo completo en la cola de origen, si el valor de USE_GROUPS_KEY es TRUE.

Los nombres de clave y valor suministrados en la Tabla 3 son constantes que se definen en la interfaz FileMetaDataConstants.

Salidas de usuario del supervisor de recursos de MFT

Las salidas de usuario del supervisor de recursos permiten configurar que el código personalizado se ejecute cuando se cumple una condición desencadenante de un supervisor antes de que se inicie la tarea asociada.

No es recomendable invocar transferencias nuevas directamente desde el código de salida de usuario. En algunas circunstancias esto hace que los archivos se transfieren varias veces porque las salidas de usuario no son resistentes a los reinicios de agente.

Las salidas de usuario del supervisor de recursos utilizan la infraestructura existente para salidas de usuario. Las salidas de usuario de supervisor se invocan después de que se haya desencadenado un supervisor, pero antes de que se haya ejecutado la tarea correspondiente mediante la tarea del supervisor. Esto permite que la salida de usuario modifique la tarea que se va a ejecutar y decida si una tarea debe continuar o no. Puede modificar la tarea de supervisor actualizando los metadatos de supervisor, que a su vez se utilizan para la sustitución de variable en el documento de la tarea creado mediante la creación del supervisor original. O bien, la salida de supervisor puede sustituir o actualizar la serie XML de definición de tarea pasada como parámetro. La salida de supervisor puede devolver un código de resultado de 'proceed' (proseguir) o 'cancel' (cancelar) para la tarea. Si se devuelve un código de resultado de cancelación, la tarea no se iniciará y el supervisor no se volverá a iniciar hasta que el recurso supervisado coincida con las condiciones desencadenantes. Si el recurso no ha cambiado, el desencadenante no se iniciará. Al igual que sucede con las otras salidas de usuario, pueden encadenarse conjuntamente salidas de usuario. Si una de las salidas devuelve un código de resultado de cancelación, se cancela el resultado global y no se inicia la tarea.

- Una correlación de metadatos de entorno (lo mismo que las otras salidas de usuario)
- Una correlación de metadatos de supervisor incluidos los metadatos del sistema inmutables y los metadatos del sistema mutables. Los metadatos del sistema inmutables son los siguientes:
  - FILENAME - nombre del archivo que cumplía la condición desencadenante
  - FILEPATH - vía de acceso al archivo que cumplía la condición desencadenante
  - FILESIZE (en bytes - puede que estos metadatos no estén disponibles) - tamaño del archivo que cumplía la condición desencadenante
  - LASTMODIFIEDDATE (Local) - fecha en la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta fecha se expresa como fecha local del huso horario en el que se ejecuta el agente y se formatea como fecha ISO 8601.
  - LASTMODIFIEDTIME (Local) - hora, en formato local, a la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta hora se expresa como hora local del huso horario en el que se ejecuta el agente y tiene un formato como el de la hora ISO 8601.
  - LASTMODIFIEDDATEUTC - fecha, en formato universal, en la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta fecha se expresa como fecha local convertida al huso horario UTC y se formatea como fecha ISO 8601.
  - LASTMODIFIEDTIMEUTC - hora, en formato universal, a la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta hora se expresa como la hora local convertida al huso horario UTC y tiene un formato de hora ISO 8601.
  - AGENTNAME - el nombre del agente supervisor
- Una serie XML que representa la tarea que se va a ejecutar como resultado del desencadenante de supervisor.

Las salidas de supervisor devuelven los siguientes datos:

- Un indicador que especifica si se debe seguir avanzando (proseguir o cancelar)
- Una serie para insertar en el mensaje de registro que ha cumplido el desencadenante

Como resultado de la ejecución del código de salida de supervisor, puede que también se hayan actualizado los metadatos del supervisor y la serie XML de definición de tarea que se transfirieron originalmente como parámetros.

El valor de la propiedad de agente monitorExitClasses (en el archivo `agent.properties`) especifica qué clases de salida de supervisor se deben cargar; cada clase de salida va separada por una coma. Por ejemplo:

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

La interfaz con la salida de usuario de supervisor es la siguiente:

```
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);
}
```

Las constantes de los valores reservados de IBM en los metadatos del supervisor son las siguientes:

```
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";

}
```

Ejemplo de rutina de salida de usuario

En este ejemplo de clase se implementa la interfaz MonitorExit. Este ejemplo añade una variable de sustitución personalizada a los metadatos de supervisor denominados *REDIRECTEDAGENT* que se llenarán con un valor de `LONDON` si la hora del día es impar y un valor de `PARIS` para horas pares. El código de resultado de salida de supervisor está definido para devolver siempre un valor `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;
    }
}
```

La tarea correspondiente para un supervisor que emplea la variable de sustitución *REDIRECTEDAGENT* podría ser similar a la siguiente:

```
<?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>
```

Antes de que se inicie esta transferencia, el valor del atributo `agent` del elemento <destinationAgent> se sustituye por `LONDON` o `PARIS`.

Debe especificar la variable de sustitución en la clase de salida del supervisor y la serie XML de definición de tarea en mayúsculas.

Conceptos relacionados

“Metadatos para salidas de usuario de MFT” en la página 2181
Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

“Interfaces de Java para salidas de usuario de MFT” en la página 2191
Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Salidas de usuario de origen y destino de MFT

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

"Propiedades de agente MFT para salidas de usuario" en la página 2189

Además de las propiedades estándar del archivo `agent.properties`, hay varias propiedades avanzadas específicas para las rutinas de salida de usuario. Estas propiedades no están incluidas de forma predeterminada; por tanto, si desea utilizar alguna de ellas, deberá editar manualmente el archivo `agent.properties`. Si realiza algún cambio en el archivo `agent.properties` mientras se ejecuta dicho agente, detenga y vuelva a iniciar el agente para recuperar los cambios.

Propiedades de agente MFT para salidas de usuario

Además de las propiedades estándar del archivo `agent.properties`, hay varias propiedades avanzadas específicas para las rutinas de salida de usuario. Estas propiedades no están incluidas de forma predeterminada; por tanto, si desea utilizar alguna de ellas, deberá editar manualmente el archivo `agent.properties`. Si realiza algún cambio en el archivo `agent.properties` mientras se ejecuta dicho agente, detenga y vuelva a iniciar el agente para recuperar los cambios.

Para IBM WebSphere MQ 7.5 o posterior, las variables de entorno se pueden utilizar en algunas propiedades de Managed File Transfer que representan ubicaciones de archivo o directorio. Esto permite que las ubicaciones de archivos o directorios que se utilizan al ejecutar componentes del producto, varíen dependiendo de los cambios del entorno, como por ejemplo qué usuario está ejecutando el proceso. Para obtener más información, consulte Variables de entorno en propiedades de MFT.

Propiedades de rutina de salida de usuario

Las rutinas de salida de usuario se llaman en el orden listado en la tabla siguiente. Para obtener más información sobre el archivo `agent.properties` , consulte Propiedades avanzadas del agente: rutina de salida de usuario.

Tabla 886. Propiedades de agente para salidas de usuario

Nombre de propiedad	Descripción
sourceTransferEndExitClasses	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de finalización de transferencia de origen.
sourceTransferStartExitClasses	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de inicio de transferencia de origen.
destinationTransferStartExitClasses	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de usuario de inicio de transferencia.
destinationTransferEndExitClasses	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de usuario de transferencia de destino.
exitClassPath	Especifica una lista de directorios, delimitados por caracteres y específicos de la plataforma, que actúan como la vía de acceso de clases para rutinas de salida de usuario. El directorio de salidas de agente se busca antes que las entradas de esta vía de acceso de clases. Si está utilizando esta propiedad en Windows, utilice un carácter de barra inclinada (/) como delimitador de vía de acceso, no el carácter de barra inclinada invertida (\). Por ejemplo: `exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.` Para IBM WebSphere MQ 7.5 o posterior, el valor de esta propiedad puede contener variables de entorno.

Tabla 886. Propiedades de agente para salidas de usuario (continuación)	
Nombre de propiedad	**Descripción**
exitNativeLibraryPath	Especifica una lista de directorios, delimitados por caracteres y específicos de la plataforma, que funcionan como vía de acceso de biblioteca nativa para rutinas de salida de usuario. Para IBM WebSphere MQ 7.5 o posterior, el valor de esta propiedad puede contener variables de entorno.
monitorExitClasses	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de supervisor. Para obtener más información, consulte Salidas de usuario del supervisor de recursos de MFT.
protocolBridgeCredentialExitClasses	Especifica una lista de clases separadas por comas que implementan una rutina de salida de usuario de credenciales de puente de protocolo. Para obtener más información, consulte Correlacionar credenciales para un servidor de archivos utilizando clases de salida.
protocolBridgePropertiesExitClasses	Especifica una lista separada por comas de clases que implementan una rutina de salida de usuario de las propiedades de servidor de puente de protocolo. Para obtener más información, consulte ProtocolBridgePropertiesExit2: buscar propiedades de servidor de archivos de protocolo.
IOExitClasses	Especifica una lista separada por comas de clases que implementan una rutina de salida de usuario de E/S. Especifique sólo las clases que implementan la interfaz IOExit, es decir, no especifique clases que implementan las otras interfaces de salida de usuario de E/S, por ejemplo IOExitResourcePath e IOExitChannel. Para obtener más información, consulte Utilización de salidas de usuario de E/S de transferencia de MFT.

Orden de invocación de salida

Las salidas de origen y de destino se invocan en el orden siguiente:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Encadenamiento de salidas de origen y de destino

Si se especifican varias salidas, la primera salida de la lista se invoca primero, seguida de la segunda salida y así sucesivamente. Los cambios realizados por la primera salida se transfieren como entrada a la salida que se invoque posteriormente y así sucesivamente. Por ejemplo, si hay dos salidas de inicio de transferencia de origen, los cambios realizados en los metadatos de la transferencia por la primera salida se entran en la segunda salida. Cada salida devuelve su propio resultado. Si todas las salidas de un tipo determinado devuelven PROCEED como código de resultado de salida, el resultado global será PROCEED. Si una o varias salidas devuelven CANCEL_TRANSFER, el resultado global será CANCEL_TRANSFER. Todos los códigos de resultado y series devuelvas por las salidas se generan como salida en el registro de transferencias.

Si el resultado global de la salida de inicio de la transferencia de origen es PROCEED, la transferencia proseguirá utilizando los cambios realizados por las salidas. Si el resultado global es CANCEL_TRANSFER, se invocarán las salidas de finalización de la transferencia de origen y se cancelará la transferencia. El estado de terminación en el registro de transferencias es "cancelado".

Si el resultado global de las salidas de inicio de la transferencia de destino es PROCEED, la transferencia proseguirá utilizando los cambios realizados por las salidas. Si el resultado global es CANCEL_TRANSFER, se invocarán las salidas de finalización de la transferencia de destino y a continuación, las salidas de finalización de la transferencia de origen. Finalmente, se cancelará la transferencia. El estado de terminación en el registro de transferencias es "cancelado".

Si una salida de origen o destino necesita transferir información a las siguientes salidas en la cadena o en el orden de ejecución, deberá realizarlo actualizando los metadatos de la transferencia. El uso

de los metadatos de transferencia es específico de la implementación de salida. Por ejemplo, si una salida establece el resultado de retorno en CANCEL_TRANSFER y necesita comunicarse con las siguientes salidas que la transferencia ha cancelado, deberá realizarlo estableciendo un valor de metadatos de transferencia que sea comprensible para las otras salidas.

Ejemplo

```
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
```

Conceptos relacionados

Personalización de MFT con salidas de usuario

“Metadatos para salidas de usuario de MFT” en la página 2181

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

“Interfaces de Java para salidas de usuario de MFT” en la página 2191

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Referencia relacionada

“Salidas de usuario del supervisor de recursos de MFT” en la página 2185

Las salidas de usuario del supervisor de recursos permiten configurar que el código personalizado se ejecute cuando se cumple una condición desencadenante de un supervisor antes de que se inicie la tarea asociada.

Variables de entorno en propiedades MFT

El archivo MFT `agent.properties`

Interfaces de Java para salidas de usuario de MFT

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Interfaz 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);      }
```

Interfaz 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;
    }
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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);
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz SourceTransferEndExit.java” en la página 2218

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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);
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz SourceTransferEndExit.java” en la página 2218

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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();
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz DestinationTransferStartExit.java” en la página 2194
“Interfaz DestinationTransferEndExit.java” en la página 2193
“Interfaz MonitorExit.java” en la página 2213
“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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;
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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;
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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();
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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();
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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;
    }
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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;
```

```
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

z/OS *Interfaz 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;
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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();

}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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 {
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz 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);
 }
```

Tareas relacionadas

Supervisión de recursos de MFT

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz SourceTransferEndExit.java” en la página 2218

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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);

}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Correlacionar credenciales para un servidor de archivos utilizando clases de salida

Interfaz 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);
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Correlacionar credenciales para un servidor de archivos utilizando clases de salida

Interfaz 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);

}
```

Tareas relacionadas

ProtocolBridgePropertiesExit: buscar propiedades de servidor de archivos de protocolo

Personalización de MFT con salidas de usuario

Correlacionar credenciales para un servidor de archivos utilizando clases de salida

Clase SourceFileExitFileSpecification.java

SourceFileExitFileSpecification.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2012, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor.  Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *            the source file specification to associate with the source file
     *            exit file specification.
     *
     * @param destinationFileSpecification
     *            the destination file specification to associate with the
     *            source file exit file specification.
     *
     * @param sourceFileMetaData
     *            the source file meta data.
     *
     * @param destinationFileMetaData
     *            the destination file meta data   .
     */
    public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                           final String destinationFileSpecification,
                                           final Map<String, String> sourceFileMetaData,
                                           final Map<String, String> destinationFileMetaData) {
        this.sourceFileSpecification = sourceFileSpecification;
        this.destinationFileSpecification = destinationFileSpecification;
        this.sourceFileMetaData = sourceFileMetaData;
        this.destinationFileMetaData = destinationFileMetaData;
    }

    /**
     * Returns the destination file specification.
     *
     * @return    the destination file specification.  This represents the location,
```

```
     *            on the agent acting as the destination for the transfer, where the
     *            file should be written.  Exit routines installed into the agent
     *            acting as the destination for the transfer may override this value.
     */
    public String getDestination() {
        return destinationFileSpecification;
    }

    /**
     * Returns the source file specification.
     *
     * @return    the source file specification.  This represents the location where
     *            the file data will be read from.
     */
    public String getSource() {
        return sourceFileSpecification;
    }

    /**
     * Returns the file meta data that relates to the source file specification.
     *
     * @return the file meta data that relates to the source file specification.
     */
    public Map<String, String> getSourceFileMetaData() {
        return sourceFileMetaData;
    }

    /**
     * Returns the file meta data that relates to the destination file specification.
     *
     * @return the file meta data that relates to the destination file specification.
     */
    public Map<String, String> getDestinationFileMetaData() {
        return destinationFileMetaData;
    }
}
```

Conceptos relacionados

“Metadatos para salidas de usuario de MFT” en la página 2181

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

Interfaz 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);


}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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);
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Clase SourceFileExitFileSpecification.java” en la página 2217

“Interfaz SourceTransferEndExit.java” en la página 2218

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Interfaz 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;
    }
}
```

Tareas relacionadas

Personalización de MFT con salidas de usuario

Referencia relacionada

“Interfaz SourceTransferStartExit.java” en la página 2219

“Interfaz DestinationTransferStartExit.java” en la página 2194

“Interfaz DestinationTransferEndExit.java” en la página 2193

“Interfaz MonitorExit.java” en la página 2213

“Interfaz ProtocolBridgeCredentialExit.java” en la página 2214

Formatos de mensaje para los mensajes que puede colocar en la cola de mandatos del agente de MFT

Estos esquemas XML definen los formatos de los mensajes que se pueden colocar en la cola de mandatos del agente para solicitar que el agente realice una acción. El mensaje XML se puede colocar en la cola de mandatos de agente mediante los mandatos de una línea de mandato o mediante un aplicación.

- Formato de mensaje de solicitud de transferencia de archivos
- Formatos de mensajes de solicitud del supervisor de MFT
- Formato de mensaje de solicitud de ping a un agente MFT
- Formato de mensaje de respuesta del agente MFT

V 9.1.0 Referencia de la REST API de mensajería

Información de referencia sobre messaging REST API.

Para obtener más información sobre cómo utilizar messaging REST API, consulte Mensajería utilizando REST API.

V 9.1.0 Recursos de REST API

Esta colección de temas proporciona información de referencia para cada uno de los recursos de messaging REST API .

Para obtener más información sobre cómo utilizar messaging REST API, consulte Mensajería utilizando REST API.

V 9.1.0 /messaging/qmgr/{qmgrName}/queue/{queueName}/message

La API REST de mensajería permite que los mensajes se coloquen en una cola, V 9.1.3 o que los mensajes se examinen o se obtengan de forma destructiva de una cola, utilizando el recurso `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` .

V 9.1.0 *POST*

Puede utilizar el método HTTP POST con el recurso `/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message` para transferir mensajes a la cola especificada en el gestor de colas especificado.

Coloca un mensaje IBM MQ que contiene el cuerpo de solicitud HTTP en el gestor de colas y cola especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El método sólo da soporte a cuerpos de solicitud HTTP basados en texto. Los mensajes se envían como mensajes con formato `MQSTR` y se colocan utilizando el contexto de usuario actual.

- “URL de recurso” en la página 2223
- “Cabeceras de solicitud” en la página 2223
- “Formato de cuerpo de solicitud” en la página 2224
- “Requisitos de seguridad” en la página 2224
- “Códigos de estado de respuesta” en la página 2225
- “Cabeceras de respuesta” en la página 2225
- “Formato de cuerpo de respuesta” en la página 2226
- “Ejemplos” en la página 2226

URL de recurso

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

Nota: Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, debe sustituir v1 donde el URL utiliza v2. Por ejemplo, la primera parte del URL es la siguiente: `https://host:port/ibmmq/rest/v1/`

qmgrName
Especifica el nombre del gestor de colas al que se debe conectar para la mensajería. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada se debe codificar como %2F.
- Un punto se debe codificar como %2E.
- Un signo de porcentaje se debe codificar como %25.

queueName
Especifica el nombre de la cola en la que colocar el mensaje.

La cola debe definirse como local, remota o un alias para el gestor de colas especificado; también puede hacer referencia a una cola en clúster.

El nombre de cola distingue entre mayúsculas y minúsculas.

Si el nombre de cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben estar codificados en URL:

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte Configuración de puertos HTTP y HTTPS.

Cabeceras de solicitud

Las cabeceras siguientes deben enviarse con la solicitud:

Autorización
Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte Utilización de la autenticación básica HTTP con REST API.

Tipo de contenido
Esta cabecera debe enviarse con uno de los valores siguientes:

- `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`

Nota: Si se omite *charset* en la cabecera `Context-Type` , se presupone `UTF-8` .

ibm-mq-rest-csrf-token
Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Las cabeceras siguientes se pueden enviar opcionalmente con la solicitud:

Accept-Language
Esta cabecera especifica el idioma necesario para las excepciones o mensajes de error devueltos en el cuerpo del mensaje de respuesta.

ibm-mq-md-correlationId
Esta cabecera establece el ID de correlación del mensaje creado. La cabecera debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-correlationId: 414d5120514d414445562020202020202067d8bf5923582e02
```

ibm-mq-md-caducidad
Esta cabecera establece la duración de caducidad para el mensaje creado. La caducidad de un mensaje se inicia desde el momento en que el mensaje llega a la cola. Como resultado, se ignora la latencia de red. La cabecera debe especificarse como uno de los valores siguientes:

ilimitado
El mensaje no caduca.

Este es el valor predeterminado.

Valor entero
Milisegundos antes de que caduque el mensaje.

Limitado al rango de 0 a 99999999900.

ibm-mq-md-persistence
Esta cabecera establece la persistencia para el mensaje creado. La cabecera debe especificarse como uno de los valores siguientes:

nonPersistent
El mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

Este es el valor predeterminado.

persistentes
El mensaje sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

ibm-mq-md-replyTo
Esta cabecera establece el destino de respuesta para el mensaje creado. El formato de la cabecera utiliza la notación estándar de proporcionar la cola de respuesta y un gestor de colas opcional: `replyQueue[@replyQmgr]`

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

Formato de cuerpo de solicitud

El cuerpo de la solicitud debe ser texto y utilizar la codificación UTF-8 . No es necesaria ninguna estructura de texto específica. Se crea un mensaje con formato `MQSTR` que contiene el texto del cuerpo de la solicitud y se coloca en la cola especificada.

Para obtener más información, consulte ejemplos.

Requisitos de seguridad

El interlocutor debe estar autenticado en el servidor mqweb. Los roles `MQWebAdmin` y `MQWebAdminRO` no son aplicables para messaging REST API. Para obtener más información sobre la seguridad de REST API, consulte Seguridad de IBM MQ Console y REST API.

Una vez autenticado en el servidor mqweb, el usuario puede utilizar messaging REST API y administrative REST API.

Al principal de seguridad del interlocutor se le debe otorgar la capacidad de transferir mensajes a la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para PUT.

- ULW MQ Appliance Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +PUT al principal de seguridad del interlocutor.
- z/OS Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar acceso UPDATE al principal de seguridad del interlocutor.

ULW En UNIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (otorgar o revocar autorización).

z/OS En z/OS, consulte Configuración de la seguridad en z/OS.

Si utiliza Advanced Message Security (AMS) con la messaging REST API, tenga en cuenta que todos los mensajes se cifran utilizando el contexto del servidor mqweb, no el contexto del usuario que publica el mensaje.

Códigos de estado de respuesta

201
El mensaje se ha creado y enviado correctamente.

400
Se han proporcionado datos no válidos.

Por ejemplo, se ha especificado un valor de cabecera de solicitud no válido.

401
No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. También se debe especificar la cabecera `ibm-mq-rest-csrf-token` . Para obtener más información, consulte “Requisitos de seguridad” en la página 2224.

403
No autorizado.

El interlocutor se autentica en el servidor mqweb y está asociado con un principal válido. Sin embargo, el principal no tiene acceso a todos o a un subconjunto de los recursos de IBM MQ necesarios, o no está en el rol `MQWebUser` . Para obtener más información sobre el acceso necesario, consulte “Requisitos de seguridad” en la página 2224.

404
No existe la cola.

405
La cola está inhibida por PUT.

415
Una cabecera o cuerpo de mensaje es un tipo de soporte no soportado.

Por ejemplo, la cabecera `Content-Type` se establece en un tipo de soporte no soportado.

500
Problema de servidor o código de error de IBM MQ.

502
El principal de seguridad actual no puede enviar el mensaje porque el proveedor de mensajería no da soporte a la función necesaria. Por ejemplo, si la vía de acceso de clases del servidor mqweb no es válida.

503
El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Las cabeceras siguientes se devuelven con la respuesta:

Contenido-Idioma
Especifica el identificador de idioma del mensaje de respuesta en caso de errores o excepciones. Se utiliza junto con la cabecera de solicitud `Accept-Language` para indicar el idioma necesario para cualquier condición de error o excepción. El valor predeterminado del servidor mqweb se utiliza si el idioma solicitado no está soportado.

Content-Length
Especifica la longitud del cuerpo de respuesta HTTP, incluso cuando no hay contenido. Tras el éxito, el valor es cero.

Tipo de contenido
Especifica el tipo de cuerpo de respuesta. Tras el éxito, el valor es `text/plain;charset=utf-8`. En el caso de errores o excepciones, el valor es `application/json;charset=utf-8`.

ibm-mq-md-messageId
Especifica el ID de mensaje asignado por IBM MQ a este mensaje. Al igual que la cabecera de solicitud `ibm-mq-md-correlationId` , se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020202067d8ce5923582f07
```

Nota: La prioridad de mensaje predeterminada para POST es 4.

Formato de cuerpo de respuesta

El cuerpo de respuesta está vacío si el mensaje se envía correctamente. Si se produce un error, el cuerpo de respuesta contiene un mensaje de error. Para obtener más información, consulte Manejo de errores deREST API.

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

El ejemplo siguiente inicia sesión en un usuario denominado `mquser` con la contraseña `mquser`. En cURL, la solicitud de inicio de sesión puede ser similar al ejemplo siguiente de Windows . La señal LTPA se almacena en el archivo `cookiejar.txt` utilizando el distintivo `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, la señal LTPA y la cabecera HTTP `ibm-mq-rest-csrf-token` se utilizan para autenticar más solicitudes. El `ibm-mq-rest-csrf-token` `token_value` puede ser cualquier valor, incluido en blanco.

- El siguiente ejemplo de Windows cURL envía un mensaje a la cola `Q1` en el gestor de colas `QM1`, utilizando las opciones predeterminadas. El mensaje contiene el texto *"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!"
  ```

- El siguiente ejemplo de Windows cURL envía un mensaje persistente a la cola `Q1` en el gestor de colas `QM1`, con una caducidad de 2 minutos. El mensaje contiene el texto *"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!"
  ```

- El siguiente ejemplo de Windows cURL envía un mensaje no persistente a la cola Q1 en el gestor de colas QM1, sin caducidad ni ID de correlación definido. El mensaje contiene el texto *"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!"
```

V 9.1.3 *GET*

V 9.1.3 Puede utilizar el método HTTP GET con el recurso `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` para examinar mensajes del gestor de colas y la cola asociados.

Examina el primer mensaje disponible del gestor de colas y la cola especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El cuerpo del mensaje se devuelve en el cuerpo de respuesta HTTP. El mensaje debe tener un formato de MQSTR y se recibe utilizando el contexto de usuario actual.

Todos los mensajes se dejan en la cola y se devuelve un código de estado adecuado al interlocutor para los mensajes inadecuados. Por ejemplo, un mensaje que no tiene un formato MQSTR .

- "URL de recurso" en la página 2227
- "Parámetros de consulta opcionales" en la página 2228
- "Cabeceras de solicitud" en la página 2228
- "Formato de cuerpo de solicitud" en la página 2228
- "Requisitos de seguridad" en la página 2228
- "Códigos de estado de respuesta" en la página 2229
- "Cabeceras de respuesta" en la página 2230
- "Formato de cuerpo de respuesta" en la página 2231
- "Ejemplos" en la página 2231

URL de recurso

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

Nota: Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, debe sustituir v1 donde el URL utiliza v2. Por ejemplo, la primera parte del URL es la siguiente: `https://host:port/ibmmq/rest/v1/`

qmgrName

Especifica el nombre del gestor de colas al que se debe conectar para la mensajería. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName

Especifica el nombre de la cola desde la que examinar el mensaje.

La cola debe definirse como local o un alias que apunte a una cola local.

El nombre de cola distingue entre mayúsculas y minúsculas.

Si el nombre de cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben estar codificados en URL:

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte Configuración de puertos HTTP y HTTPS.

Parámetros de consulta opcionales

correlationId=*hexValue*
Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue
El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?correlationId=414d5120514d414445562020202020202067d8bf5923582e02
```

messageId=*hexValue*
Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue
El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?messageId=414d5120514d414445562020202020202067d8ce5923582f07
```

Cabeceras de solicitud

Las cabeceras siguientes deben enviarse con la solicitud:

Autorización
Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte Utilización de la autenticación básica HTTP con REST API.

ibm-mq-rest-csrf-token
Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Las cabeceras siguientes se pueden enviar opcionalmente con la solicitud:

Aceptar-Juego de caracteres
Esta cabecera se puede utilizar para indicar qué juego de caracteres es aceptable para la respuesta. Si se especifica, esta cabecera se debe establecer como UTF-8.

Accept-Language
Esta cabecera especifica el idioma necesario para las excepciones o mensajes de error devueltos en el cuerpo del mensaje de respuesta.

Formato de cuerpo de solicitud

Ninguno.

Requisitos de seguridad

El interlocutor debe estar autenticado en el servidor mqweb. Los roles `MQWebAdmin` y `MQWebAdminRO` no son aplicables para messaging REST API. Para obtener más información sobre la seguridad de REST API, consulte Seguridad de IBM MQ Console y REST API.

Una vez autenticado en el servidor mqweb, el usuario puede utilizar messaging REST API y administrative REST API.

Se debe otorgar al principal de seguridad del proceso de llamada la capacidad de examinar mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para BROWSE.
- ULW MQ Appliance Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
- z/OS Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

ULW En UNIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (otorgar o revocar autorización).

z/OS En z/OS, consulte Configuración de la seguridad en z/OS.

Códigos de estado de respuesta

200
El mensaje se ha recibido correctamente.

204
No hay mensajes disponibles.

400
Se han proporcionado datos no válidos.

Por ejemplo, se ha especificado un valor de parámetro de consulta no válido.

401
No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar la cabecera ibm-mq-rest-csrf-token . Para obtener más información, consulte “Requisitos de seguridad” en la página 2228.

403
No autorizado.

El interlocutor se autentica en el servidor mqweb y está asociado con un principal válido. Sin embargo, el principal no tiene acceso a todos o a un subconjunto de los recursos de IBM MQ necesarios, o no está en el rol MQWebUser . Para obtener más información sobre el acceso necesario, consulte “Requisitos de seguridad” en la página 2228.

404
No existe la cola.

500
Problema de servidor o código de error de IBM MQ.

501
No se ha podido construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto, pero el cuerpo no se ha podido procesar.

502
El principal de seguridad actual no puede recibir el mensaje porque el proveedor de mensajería no da soporte a la función necesaria. Por ejemplo, si la vía de acceso de clases del servidor mqweb no es válida.

503
El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Las cabeceras siguientes se devuelven con la respuesta:

Contenido-Idioma
Especifica el identificador de idioma del mensaje de respuesta en caso de errores o excepciones. Se utiliza junto con la cabecera de solicitud `Accept-Language` para indicar el idioma necesario para cualquier condición de error o excepción. El valor predeterminado del servidor mqweb se utiliza si el idioma solicitado no está soportado.

Content-Length
Especifica la longitud del cuerpo de respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud (bytes) de los datos del mensaje.

Tipo de contenido
Especifica el tipo de contenido devuelto en el cuerpo de respuesta del mensaje recibido. Tras el éxito, el valor es `text/plain;charset=utf-8`. En el caso de errores o excepciones, el valor es `application/json;charset=utf-8`.

ibm-mq-md-correlationId
Especifica el ID de correlación del mensaje recibido. La cabecera se devuelve si el mensaje recibido contiene un ID de correlación válido. Se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-correlationId: 414d5120514d414445562020202020202067d8bf5923582e02
```

ibm-mq-md-caducidad
Especifica la duración de caducidad restante del mensaje recibido. La cabecera puede ser uno de los valores siguientes:

ilimitado
El mensaje no caduca.

Valor entero
Milisegundos restantes antes de que caduque el mensaje.

ibm-mq-md-messageId
Especifica el ID de mensaje asignado por IBM MQ a este mensaje. Al igual que la cabecera `ibm-mq-md-correlationId` , se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020202067d8ce5923582f07
```

ibm-mq-md-persistence
Especifica la persistencia del mensaje recibido. La cabecera puede ser uno de los valores siguientes:

nonPersistent
El mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

persistentes
El mensaje sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

ibm-mq-md-replyTo
Especifica el destino de respuesta para el mensaje recibido. El formato de la cabecera utiliza la notación estándar de la cola de respuesta y el gestor de colas, `replyQueue@replyQmgr`.

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

Formato de cuerpo de respuesta

Tras el éxito, el cuerpo de la respuesta contiene el cuerpo del mensaje del mensaje recibido. Si se produce un error, el cuerpo de respuesta contiene un mensaje de error con formato JSON. Ambas respuestas están codificadas en UTF-8 . Para obtener más información, consulte Manejo de errores deREST API.

Tenga en cuenta que al recibir un mensaje solo se admiten los mensajes con formato IBM MQ MQSTR .

Examinar una cola que se ha marcado como GET inhibido no devuelve ningún contenido.

Si la cola que se está examinando contiene mensajes con identificadores de mensaje duplicados, se devuelve el primer mensaje al filtrar por el identificador de mensaje.

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

El ejemplo siguiente inicia sesión en un usuario denominado `mquser` con la contraseña `mquser`. En cURL, la solicitud de inicio de sesión puede ser similar al ejemplo siguiente de Windows . La señal LTPA se almacena en el archivo `cookiejar.txt` utilizando el distintivo `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, la señal LTPA y la cabecera HTTP `ibm-mq-rest-csrf-token` se utilizan para autenticar más solicitudes. El `ibm-mq-rest-csrf-token` `token_value` puede ser cualquier valor, incluido en blanco.

- El siguiente ejemplo de Windows cURL examina el siguiente mensaje disponible de la cola `Q1` en el gestor de colas `QM1`, utilizando las opciones predeterminadas:

  ```
  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"
  ```

- El siguiente ejemplo de Windows cURL examina un mensaje con un ID de correlación específico, `000000000000000000000000000000000000000000abcdabcd`, de la cola `Q1` en el gestor de colas `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=000000000000000000000000000000000000000000abcdabcd"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

V 9.1.0 *DELETE*

Puede utilizar el método HTTP DELETE con el recurso `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` para obtener mensajes del gestor de colas y la cola asociados.

Obtiene de forma destructiva el siguiente mensaje disponible del gestor de colas y cola especificados, devolviendo el cuerpo del mensaje en el cuerpo de respuesta HTTP. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El mensaje debe tener un formato de `MQSTR`y se recibe utilizando el contexto de usuario actual.

Los mensajes incompatibles se dejan en la cola y se devuelve un código de estado adecuado al interlocutor. Por ejemplo, un mensaje que no tiene un formato `MQSTR` .

- “URL de recurso” en la página 2232
- “Parámetros de consulta opcionales” en la página 2232
- “Cabeceras de solicitud” en la página 2233

- “Formato de cuerpo de solicitud” en la página 2233
- “Requisitos de seguridad” en la página 2233
- “Códigos de estado de respuesta” en la página 2234
- “Cabeceras de respuesta” en la página 2234
- “Formato de cuerpo de respuesta” en la página 2235
- “Ejemplos” en la página 2236

URL de recurso

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

Nota: Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, debe sustituir v1 donde el URL utiliza v2. Por ejemplo, la primera parte del URL es la siguiente: `https://host:port/ibmmq/rest/v1/`

qmgrName
Especifica el nombre del gestor de colas al que se debe conectar para la mensajería. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName
Especifica el nombre de la cola de la que se obtiene el siguiente mensaje.

La cola debe definirse como local o como un alias que apunta a una cola local.

El nombre de cola distingue entre mayúsculas y minúsculas.

Si el nombre de cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben estar codificados en URL:

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte Configuración de puertos HTTP y HTTPS.

Parámetros de consulta opcionales

correlationId=*hexValue*
Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue
El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?correlationId=414d5120514d414445562020202020202067d8bf5923582e02
```

messageId=*hexValue*
Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue
El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?messageId=414d5120514d414445562020202020202067d8ce5923582f07
```

wait=*integerValue*
Especifica que el método HTTP esperará *integerValue* milisegundos a que el siguiente mensaje pase a estar disponible.

integerValue
El parámetro de consulta debe especificarse como un valor entero que representa la duración de milisegundos. El valor máximo es 2147483647.

Por ejemplo:

```
../message?wait=120000
```

Cabeceras de solicitud

Las cabeceras siguientes deben enviarse con la solicitud:

Autorización
Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte Utilización de la autenticación básica HTTP con REST API.

ibm-mq-rest-csrf-token
Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Las cabeceras siguientes se pueden enviar opcionalmente con la solicitud:

Aceptar-Juego de caracteres
Esta cabecera se puede utilizar para indicar qué juego de caracteres es aceptable para la respuesta. Si se especifica, esta cabecera se debe establecer como UTF-8.

Accept-Language
Esta cabecera especifica el idioma necesario para las excepciones o mensajes de error devueltos en el cuerpo del mensaje de respuesta.

Formato de cuerpo de solicitud

Ninguno.

Requisitos de seguridad

El interlocutor debe estar autenticado en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad de REST API, consulte Seguridad de IBM MQ Console y REST API.

Una vez autenticado en el servidor mqweb, el usuario puede utilizar messaging REST API y administrative REST API.

Al principal de seguridad del interlocutor se le debe otorgar la capacidad de obtener mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para GET.
- ULW MQ Appliance Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
- z/OS Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

ULW En UNIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (otorgar o revocar autorización).

z/OS En z/OS, consulte Configuración de la seguridad en z/OS.

Códigos de estado de respuesta

200
El mensaje se ha recibido correctamente.

204
No hay mensajes disponibles.

400
Se han proporcionado datos no válidos.

Por ejemplo, se ha especificado un valor de parámetro de consulta no válido.

401
No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. También se debe especificar la cabecera `ibm-mq-rest-csrf-token` . Para obtener más información, consulte "Requisitos de seguridad" en la página 2233.

403
No autorizado.

El interlocutor se autentica en el servidor mqweb y está asociado con un principal válido. Sin embargo, el principal no tiene acceso a todos o a un subconjunto de los recursos de IBM MQ necesarios, o no está en el rol `MQWebUser` . Para obtener más información sobre el acceso necesario, consulte "Requisitos de seguridad" en la página 2233.

404
No existe la cola.

405
La cola está inhibida por GET.

500
Problema de servidor o código de error de IBM MQ.

501
No se ha podido construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto, pero el cuerpo no se ha podido procesar.

502
El principal de seguridad actual no puede recibir el mensaje porque el proveedor de mensajería no da soporte a la función necesaria. Por ejemplo, si la vía de acceso de clases del servidor mqweb no es válida.

503
El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Las cabeceras siguientes se devuelven con la respuesta:

Contenido-Idioma
Especifica el identificador de idioma del mensaje de respuesta en caso de errores o excepciones. Se utiliza junto con la cabecera de solicitud `Accept-Language` para indicar el idioma necesario para cualquier condición de error o excepción. El valor predeterminado del servidor mqweb se utiliza si el idioma solicitado no está soportado.

Content-Length
Especifica la longitud del cuerpo de respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud (bytes) de los datos del mensaje.

Tipo de contenido
Especifica el tipo de contenido devuelto en el cuerpo de respuesta del mensaje recibido. Tras el éxito, el valor es `text/plain;charset=utf-8`. En el caso de errores o excepciones, el valor es `application/json;charset=utf-8`.

ibm-mq-md-correlationId
Especifica el ID de correlación del mensaje recibido. La cabecera se devuelve si el mensaje recibido contiene un ID de correlación válido. Se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-correlationId: 414d5120514d414445562020202020202067d8bf5923582e02
```

ibm-mq-md-caducidad
Especifica la duración de caducidad restante del mensaje recibido. La cabecera puede ser uno de los valores siguientes:

ilimitado
El mensaje no caduca.

Valor entero
Milisegundos restantes antes de que caduque el mensaje.

ibm-mq-md-messageId
Especifica el ID de mensaje asignado por IBM MQ a este mensaje. Al igual que la cabecera `ibm-mq-md-correlationId` , se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020202067d8ce5923582f07
```

ibm-mq-md-persistence
Especifica la persistencia del mensaje recibido. La cabecera puede ser uno de los valores siguientes:

nonPersistent
El mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

persistentes
El mensaje sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

ibm-mq-md-replyTo
Especifica el destino de respuesta para el mensaje recibido. El formato de la cabecera utiliza la notación estándar de la cola de respuesta y el gestor de colas, `replyQueue@replyQmgr`.

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

Formato de cuerpo de respuesta

Tras el éxito, el cuerpo de la respuesta contiene el cuerpo del mensaje del mensaje recibido. Si se produce un error, el cuerpo de respuesta contiene un mensaje de error con formato JSON. Ambas respuestas están codificadas en UTF-8 . Para obtener más información, consulte Manejo de errores deREST API.

Tenga en cuenta que al recibir un mensaje solo se admiten los mensajes con formato IBM MQ MQSTR . Posteriormente, todos los mensajes se reciben bajo punto de sincronización y todos los mensajes no gestionados se dejan en la cola. La cola de IBM MQ puede configurarse para trasladar estos mensajes con

formato incorrecto a un destino alternativo. Para obtener más información, consulte Manejo de mensajes con formato incorrecto en clases IBM MQ para JMS.

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

El ejemplo siguiente inicia sesión en un usuario denominado `mquser` con la contraseña `mquser`. En cURL, la solicitud de inicio de sesión puede ser similar al ejemplo siguiente de Windows . La señal LTPA se almacena en el archivo `cookiejar.txt` utilizando el distintivo `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, la señal LTPA y la cabecera HTTP `ibm-mq-rest-csrf-token` se utilizan para autenticar más solicitudes. El `ibm-mq-rest-csrf-token` `token_value` puede ser cualquier valor, incluido en blanco.

- El siguiente ejemplo de Windows cURL elimina el siguiente mensaje disponible de la cola `Q1` en el gestor de colas `QM1`, utilizando las opciones predeterminadas:

  ```
  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"
  ```

- El siguiente ejemplo de Windows cURL elimina un mensaje con un ID de correlación específico, `000000000000000000000000000000000000000abcdabcd`, de la cola `Q1` en el gestor de colas `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=000000000000000000000000000000000000000abcdabcd"
  -X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- El siguiente ejemplo de Windows cURL elimina un mensaje con un ID de correlación específico, `000000000000000000000000000000000000000abcdabcd`, de la cola `Q1` en el gestor de colas `QM1`, esperando un máximo de 30 segundos a que el mensaje pase a estar disponible. Si pasan 30 segundos sin que el mensaje especificado se transfiera a la cola, la llamada DELETE devuelve sin un mensaje:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=000000000000000000000000000000000000000abcdabcd&wait=30000"
  -X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

V 9.1.3 /messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist

V 9.1.3 Puede utilizar el método HTTP GET con el recurso `/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist` para obtener una lista de mensajes disponibles de la cola especificada en el gestor de colas especificado.

V 9.1.3 *GET*

V 9.1.3 Puede utilizar el método HTTP GET con el recurso `/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist` para obtener una lista de mensajes disponibles de la cola especificada en el gestor de colas especificado.

Examina una lista de resumen de mensajes del gestor de colas y cola especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb. Los datos de resumen se devuelven en el cuerpo de respuesta HTTP como una matriz con formato JSON. Los datos no contienen la carga útil de los

mensajes y se reciben utilizando el contexto de usuario actual. No se elimina ningún mensaje de la cola asociada.

Si se realiza una solicitud para obtener una lista de mensajes disponibles de una cola que está inhibida GET, se devuelve una matriz JSON vacía.

- "URL de recurso" en la página 2237
- "Parámetros de consulta opcionales" en la página 2237
- "Cabeceras de solicitud" en la página 2238
- "Formato de cuerpo de solicitud" en la página 2238
- "Requisitos de seguridad" en la página 2238
- "Códigos de estado de respuesta" en la página 2239
- "Cabeceras de respuesta" en la página 2240
- "Formato de cuerpo de respuesta" en la página 2240
- "Ejemplos" en la página 2240

URL de recurso

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

Nota: Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, debe sustituir v1 donde el URL utiliza v2. Por ejemplo, la primera parte del URL es la siguiente: `https://host:port/ibmmq/rest/v1/`

qmgrName

Especifica el nombre del gestor de colas al que se debe conectar para la mensajería. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName

Especifica el nombre de la cola desde la que examinar los mensajes.

La cola debe definirse como local o un alias que apunte a una cola local.

El nombre de cola distingue entre mayúsculas y minúsculas.

Si el nombre de cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben estar codificados en URL:

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte Configuración de puertos HTTP y HTTPS.

Parámetros de consulta opcionales

correlationId=*hexValue*

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../messagelist?correlationId=414d5120514d414445562020202020202067d8bf5923582e02
```

messageId=*hexValue*
Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue
El parámetro de consulta debe especificarse como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../messagelist?messageId=414d5120514d414445562020202020202067d8ce5923582f07
```

limit=*integerValue*
Especifica que el cuerpo de respuesta del método HTTP está limitado a elementos JSON *integerValue* .

integerValue
El parámetro de consulta debe especificarse como un valor entero que representa el número máximo de elementos contenidos en el cuerpo de respuesta JSON.

El valor predeterminado es 10 y el valor máximo es 2147483647.

Por ejemplo:

```
../messagelist?limit=250
```

Cabeceras de solicitud

Las cabeceras siguientes deben enviarse con la solicitud:

Autorización
Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte Utilización de la autenticación básica HTTP con REST API.

ibm-mq-rest-csrf-token
Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Las cabeceras siguientes se pueden enviar opcionalmente con la solicitud:

Aceptar-Juego de caracteres
Esta cabecera se puede utilizar para indicar qué juego de caracteres es aceptable para la respuesta. Si se especifica, esta cabecera se debe establecer como `UTF-8`.

Accept-Language
Esta cabecera especifica el idioma necesario para las excepciones o mensajes de error devueltos en el cuerpo del mensaje de respuesta.

Formato de cuerpo de solicitud

Ninguno.

Requisitos de seguridad

El interlocutor debe estar autenticado en el servidor mqweb. Los roles `MQWebAdmin` y `MQWebAdminRO` no son aplicables para messaging REST API. Para obtener más información sobre la seguridad de REST API, consulte Seguridad de IBM MQ Console y REST API.

Una vez autenticado en el servidor mqweb, el usuario puede utilizar messaging REST API y administrative REST API.

Se debe otorgar al principal de seguridad del proceso de llamada la capacidad de examinar mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para BROWSE.
- ULW MQ Appliance Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
- z/OS Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

ULW En UNIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (otorgar o revocar autorización).

z/OS En z/OS, consulte Configuración de la seguridad en z/OS.

Códigos de estado de respuesta

200
La lista de mensajes se ha recibido correctamente.

400
Se han proporcionado datos no válidos.

Por ejemplo, se ha especificado un valor de parámetro de consulta no válido.

401
No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar la cabecera ibm-mq-rest-csrf-token . Para obtener más información, consulte “Requisitos de seguridad” en la página 2238.

403
No autorizado.

El interlocutor se autentica en el servidor mqweb y está asociado con un principal válido. Sin embargo, el principal no tiene acceso a todos o a un subconjunto de los recursos de IBM MQ necesarios, o no está en el rol MQWebUser . Para obtener más información sobre el acceso necesario, consulte “Requisitos de seguridad” en la página 2238.

404
No existe la cola.

500
Problema de servidor o código de error de IBM MQ.

501
No se ha podido construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto, pero el cuerpo no se ha podido procesar.

502
El principal de seguridad actual no puede recibir el mensaje porque el proveedor de mensajería no da soporte a la función necesaria. Por ejemplo, si la vía de acceso de clases del servidor mqweb no es válida.

503
El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Contenido-Idioma
Especifica el identificador de idioma del mensaje de respuesta en caso de errores o excepciones. Se utiliza junto con la cabecera de solicitud `Accept-Language` para indicar el idioma necesario para cualquier condición de error o excepción. El valor predeterminado del servidor mqweb se utiliza si el idioma solicitado no está soportado.

Content-Length
Especifica la longitud del cuerpo de respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud de los datos del mensaje, en bytes.

Tipo de contenido
Especifica el tipo de cuerpo de respuesta. El valor es `application/json;charset=utf-8`.

ibm-mq-total-browse-size
V 9.1.5 A partir de IBM MQ 9.1.5, esta cabecera de respuesta ya no se devuelve.

Especifica el número total de mensajes que están disponibles en la cola. Si se especifican criterios de filtrado, el número total de mensajes es el número de mensajes de la cola que coinciden con los criterios de filtrado. El valor de cabecera puede ser igual o mayor que el número de elementos JSON que se devuelven en el cuerpo de respuesta.

Formato de cuerpo de respuesta

Tras el éxito, el cuerpo de respuesta es una respuesta codificada de `UTF-8` . La respuesta contiene un objeto JSON externo que contiene una única matriz JSON denominada `messages`. Cada elemento de la matriz es un objeto JSON que contiene información sobre un mensaje en la cola. Cada elemento contiene los atributos siguientes:

correlationId
Especifica el ID de correlación del mensaje. El valor se devuelve si el mensaje contiene un ID de correlación válido. Se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

messageId
Especifica el ID de mensaje asignado por IBM MQ a este mensaje. Se representa como una serie codificada hexadecimal de 48 caracteres, que representa 24 bytes.

formato
Especifica el campo de formato MQMD. En circunstancias normales, los mensajes de texto contendrán el valor de IBM MQ `MQSTR` .

Si se realiza una solicitud para obtener una lista de mensajes en una cola que está inhibida GET, se devuelve una matriz JSON vacía.

Si se produce un error, el cuerpo de respuesta contiene un mensaje de error con formato JSON. Para obtener más información, consulte Manejo de errores deREST API.

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya `v1` donde el URL de ejemplo utiliza `v2`.

El ejemplo siguiente inicia sesión en un usuario denominado `mquser` con la contraseña `mquser`. En cURL, la solicitud de inicio de sesión puede ser similar al ejemplo siguiente de Windows . La señal LTPA se almacena en el archivo `cookiejar.txt` utilizando el distintivo `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, la señal LTPA y la cabecera HTTP `ibm-mq-rest-csrf-token` se utilizan para autenticar más solicitudes. El `ibm-mq-rest-csrf-token` `token_value` puede ser cualquier valor, incluido en blanco.

- El siguiente ejemplo de Windows cURL lista los siguientes diez mensajes disponibles de la cola `Q1` en el gestor de colas `QM1`, utilizando las opciones predeterminadas:

```
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"
```

- El siguiente ejemplo de Windows cURL lista los siguientes doscientos mensajes disponibles de la cola `Q1` en el gestor de colas `QM1`, utilizando las opciones predeterminadas:

```
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"
```

- El siguiente ejemplo de Windows cURL lista sólo los mensajes con el ID de correlación correspondiente, `0000000000000000000000000000000000000000abcdabcd`, de la cola `Q1` en el gestor de colas `QM1`:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?
correlationId=0000000000000000000000000000000000000000abcdabcd"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

V 9.1.5 /messaging/qmgr/{qmgrName}/topic/{topicString}/message

Puede utilizar el método HTTP POST con el recurso `/messaging/qmgr/{qmgrName}/topic/{topicString}/message` para publicar mensajes en el tema especificado en el gestor de colas especificado.

V 9.1.5 *POST*

Puede utilizar el método HTTP POST con el recurso `/messaging/qmgr/{qmgrName}/topic/{topicString}/message` para publicar mensajes en el tema especificado en el gestor de colas especificado.

Publica un mensaje basado en texto en el cuerpo de solicitud HTTP en el gestor de colas y tema especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb y solo se da soporte a los mensajes basados en texto. Los mensajes se publican como mensajes con formato `MQSTR` utilizando el contexto de usuario actual y tienen una prioridad de mensaje predeterminada de 4.

- “URL de recurso” en la página 2241
- “Cabeceras de solicitud” en la página 2242
- “Formato de cuerpo de solicitud” en la página 2243
- “Requisitos de seguridad” en la página 2243
- “Códigos de estado de respuesta” en la página 2244
- “Cabeceras de respuesta” en la página 2244
- “Formato de cuerpo de respuesta” en la página 2245
- “Ejemplos” en la página 2245

URL de recurso

`https://`*`host`*`:`*`port`*`/ibmmq/rest/v2/messaging/qmgr/`*`{qmgrName}`*`/topic/`*`{topicString}`*`/message`

qmgrName

Especifica el nombre del gestor de colas al que se debe conectar para la mensajería. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada se debe codificar como %2F.
- Un punto se debe codificar como %2E.
- Un signo de porcentaje se debe codificar como %25.

topicString

Especifica la serie de tema en la que publicar el mensaje.

La serie de tema distingue entre mayúsculas y minúsculas. La serie de tema puede contener varios niveles de tema, separados por el delimitador de barra inclinada.

Si la serie de tema contiene un signo de porcentaje, un punto o un signo de interrogación, estos caracteres deben estar codificados en URL:

- Un signo de porcentaje se debe codificar como %25.
- Un punto se debe codificar como %2E.
- Un signo de interrogación se debe codificar como %3F.

Si la serie de tema empieza o termina con una barra inclinada, debe codificarse con un %2F.

Por ejemplo, para publicar en la serie de tema:

- `sport/football` en el gestor de colas `MY.QMGR`, utilice el URL siguiente:

  ```
  https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/sport/football/message
  ```

- `/sport/football` en el gestor de colas `MY.QMGR`, utilice el URL siguiente:

  ```
  https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/%2Fsport/football/
  message
  ```

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte Configuración de puertos HTTP y HTTPS.

Cabeceras de solicitud

Las cabeceras siguientes deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte Utilización de la autenticación básica HTTP con REST API.

Tipo de contenido

Esta cabecera debe enviarse con uno de los valores siguientes:

- `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`

Nota: Si se omite *charset* en la cabecera `Context-Type` , se presupone `UTF-8` .

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Las cabeceras siguientes se pueden enviar opcionalmente con la solicitud:

Accept-Language
Esta cabecera especifica el idioma necesario para las excepciones o mensajes de error devueltos en el cuerpo del mensaje de respuesta.

ibm-mq-md-caducidad
Esta cabecera establece la duración de caducidad para el mensaje creado. La caducidad de un mensaje se inicia desde el momento en que el mensaje llega al gestor de colas. Como resultado, se ignora la latencia de red. La cabecera debe especificarse como uno de los valores siguientes:

ilimitado
El mensaje no caduca.

Este es el valor predeterminado.

Valor entero
Milisegundos antes de que caduque el mensaje.

Limitado al rango de 0 a 99999999900.

ibm-mq-md-persistence
Esta cabecera establece la persistencia para el mensaje creado. La cabecera debe especificarse como uno de los valores siguientes:

nonPersistent
El mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

Este es el valor predeterminado.

persistentes
El mensaje sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.

ibm-mq-md-replyTo
Esta cabecera establece el destino de respuesta para el mensaje creado. El formato de la cabecera utiliza la notación estándar de proporcionar la cola de respuesta y un gestor de colas opcional: `replyQueue[@replyQmgr]`

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

Formato de cuerpo de solicitud

El cuerpo de la solicitud debe ser texto y utilizar la codificación UTF-8 . No es necesaria ninguna estructura de texto específica. Se crea un mensaje con formato `MQSTR` que contiene el texto del cuerpo de la solicitud y se publica en el tema especificado.

Para obtener más información, consulte ejemplos.

Requisitos de seguridad

El interlocutor debe estar autenticado en el servidor mqweb. Los roles `MQWebAdmin` y `MQWebAdminRO` no son aplicables para messaging REST API. Para obtener más información sobre la seguridad de REST API, consulte Seguridad de IBM MQ Console y REST API.

Una vez autenticado en el servidor mqweb, el usuario puede utilizar messaging REST API y administrative REST API.

Al principal de seguridad del interlocutor se le debe otorgar la posibilidad de publicar mensajes en el tema especificado:

- El tema especificado por la parte *{topicString}* del URL de recurso debe estar habilitado para `PUBLISH` .
- ULW MQ Appliance Para el tema especificado por la parte *{topicString}* del URL de recurso, la autorización +PUB debe otorgarse al principal de seguridad del interlocutor.

- z/OS Para el tema especificado por la parte *{topicString}* del URL de recurso, se debe otorgar acceso UPDATE al principal de seguridad del interlocutor.

ULW En UNIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (otorgar o revocar autorización).

z/OS En z/OS, consulte Configuración de la seguridad en z/OS.

Si utiliza Advanced Message Security (AMS) con la messaging REST API, tenga en cuenta que todos los mensajes se cifran utilizando el contexto del servidor mqweb, no el contexto del usuario que publica el mensaje.

Códigos de estado de respuesta

201
El mensaje se ha creado y publicado correctamente.

400
Se han proporcionado datos no válidos.

Por ejemplo, se ha especificado un valor de cabecera de solicitud no válido.

401
No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. También se debe especificar la cabecera `ibm-mq-rest-csrf-token` . Para obtener más información, consulte "Requisitos de seguridad" en la página 2243.

403
No autorizado.

El interlocutor se autentica en el servidor mqweb y está asociado con un principal válido. Sin embargo, el principal no tiene acceso a todos o a un subconjunto de los recursos de IBM MQ necesarios, o no está en el rol `MQWebUser` . Para obtener más información sobre el acceso necesario, consulte "Requisitos de seguridad" en la página 2243.

404
El gestor de colas no existe.

405
El tema es PUBLISH inhibido.

415
Una cabecera o cuerpo de mensaje es un tipo de soporte no soportado.

Por ejemplo, la cabecera `Content-Type` se establece en un tipo de soporte no soportado.

500
Problema de servidor o código de error de IBM MQ.

502
El principal de seguridad actual no puede publicar el mensaje porque el proveedor de mensajería no da soporte a la función necesaria. Por ejemplo, si la vía de acceso de clases del servidor mqweb no es válida.

503
El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Las cabeceras siguientes se devuelven con la respuesta:

Contenido-Idioma
Especifica el identificador de idioma del mensaje de respuesta en caso de errores o excepciones. Se utiliza junto con la cabecera de solicitud `Accept-Language` para indicar el idioma necesario para cualquier condición de error o excepción. El valor predeterminado del servidor mqweb se utiliza si el idioma solicitado no está soportado.

Content-Length
Especifica la longitud del cuerpo de respuesta HTTP, incluso cuando no hay contenido. Tras el éxito, el valor es cero.

Tipo de contenido
Especifica el tipo de cuerpo de respuesta. Tras el éxito, el valor es `text/plain;charset=utf-8`. En el caso de errores o excepciones, el valor es `application/json;charset=utf-8`.

Formato de cuerpo de respuesta

El cuerpo de respuesta está vacío si el mensaje se publica correctamente. Si se produce un error, el cuerpo de respuesta contiene un mensaje de error. Para obtener más información, consulte Manejo de errores deREST API.

Ejemplos

El ejemplo siguiente inicia sesión en un usuario denominado `mquser` con la contraseña `mquser`. En cURL, la solicitud de inicio de sesión puede ser similar al ejemplo siguiente de Windows . La señal LTPA se almacena en el archivo `cookiejar.txt` utilizando el distintivo `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v1/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, la señal LTPA y la cabecera HTTP `ibm-mq-rest-csrf-token` se utilizan para autenticar más solicitudes. El `ibm-mq-rest-csrf-token` `token_value` puede ser cualquier valor, incluido en blanco.

- El siguiente ejemplo de Windows cURL publica un mensaje en la serie de tema `myTopic` en el gestor de colas `QM1`, utilizando las opciones predeterminadas. El mensaje contiene el texto *"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!"
  ```

- El siguiente ejemplo de Windows cURL publica un mensaje persistente en la serie de tema `myTopic/thisTopic` en el gestor de colas `QM1`, con una caducidad de 2 minutos. El mensaje contiene el texto *"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!"
  ```

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos.

Es posible que IBM no ofrezca los productos, servicios o las características que se tratan en este documento en otros países. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su zona. Las referencias a programas, productos o servicios de IBM no pretenden indicar ni implicar que sólo puedan utilizarse los productos, programas o servicios de IBM. En su lugar podrá utilizarse cualquier producto, programa o servicio equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio no IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran el tema principal descrito en este documento. El suministro de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director
of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe las consultas por escrito a:

Licencias de Propiedad Intelectual
Ley de Propiedad intelectual y legal
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones contradigan la legislación vigente: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN NINGÚN TIPO DE GARANTÍA, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO INCUMPLIMIENTO, COMERCIABILIDAD O IDONEIDAD PARA UNA FINALIDAD DETERMINADA. Algunas legislaciones no contemplan la exclusión de garantías, ni implícitas ni explícitas, en determinadas transacciones, por lo que puede haber usuarios a los que no les afecte dicha norma.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información aquí contenida está sometida a cambios periódicos; tales cambios se irán incorporando en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia en esta información a sitios web que no son de IBM se realiza por razones prácticas y de ninguna manera sirve como un respaldo de dichos sitios web. Los materiales de dichos sitios web no forman parte de este producto de IBM y la utilización de los mismos será por cuenta y riesgo del usuario.

IBM puede utilizar o distribuir cualquier información que el usuario le proporcione del modo que considere apropiado sin incurrir por ello en ninguna obligación con respeto al usuario.

Los titulares de licencias de este programa que deseen información del mismo con el fin de permitir: (i) el intercambio de información entre los programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N

© Copyright IBM Corp. 2007, 2024

Rochester, MN 55901
U.S.A.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluyendo, en algunos casos, el pago de una cantidad.

El programa bajo licencia que se describe en esta información y todo el material bajo licencia disponible para el mismo lo proporciona IBM bajo los términos del Acuerdo de cliente de IBM, el Acuerdo de licencia de programas internacional de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos en este documento se han obtenido en un entorno controlado. Por consiguiente, los resultados obtenidos en otros entornos operativos pueden variar de manera significativa. Es posible que algunas mediciones se hayan realizado en sistemas en nivel de desarrollo y no existe ninguna garantía de que estas mediciones serán las mismas en sistemas disponibles generalmente. Además, algunas mediciones pueden haberse estimado por extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relativa a productos que no son de IBM se obtuvo de los proveedores de esos productos, sus anuncios publicados u otras fuentes de disponibilidad pública. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad o alguna reclamación relacionada con productos que no sean de IBM. Las preguntas relacionadas con las posibilidades de los productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones relacionadas con una futura intención o tendencia de IBM están sujetas a cambios o se pueden retirar sin previo aviso y sólo representan metas y objetivos.

Este documento contiene ejemplos de datos e informes que se utilizan diariamente en la actividad de la empresa. Para ilustrar los ejemplos de la forma más completa posible, éstos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones utilizados por una empresa real es puramente casual.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin pagar ninguna cuota a IBM para fines de desarrollo, uso, marketing o distribución de programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. IBM, por tanto, no puede garantizar la fiabilidad, servicio o funciones de estos programas.

Puede que si visualiza esta información en copia software, las fotografías e ilustraciones a color no aparezcan.

Información acerca de las interfaces de programación

La información de interfaz de programación, si se proporciona, está pensada para ayudarle a crear software de aplicación para su uso con este programa.

Este manual contiene información sobre las interfaces de programación previstas que permiten al cliente escribir programas para obtener los servicios de WebSphere MQ.

Sin embargo, esta información puede contener también información de diagnóstico, modificación y ajustes. La información de diagnóstico, modificación y ajustes se proporciona para ayudarle a depurar el software de aplicación.

Importante: No utilice esta información de diagnóstico, modificación y ajuste como interfaz de programación porque está sujeta a cambios.

Marcas registradas

IBM, el logotipo de IBM , ibm.com, son marcas registradas de IBM Corporation, registradas en muchas jurisdicciones de todo el mundo. Hay disponible una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas.

Microsoft y Windows son marcas registradas de Microsoft Corporation en EE.UU. y/o en otros países.

UNIX es una marca registrada de Open Group en Estados Unidos y en otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y en otros países.

Este producto incluye software desarrollado por Eclipse Project (http://www.eclipse.org/).

Java y todas las marcas registradas y logotipos son marcas registradas de Oracle o sus afiliados.

Número Pieza:

(1P) P/N: