7.5

IBM WebSphere MQ Anwendungsreferenz entwickeln

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen” auf Seite 1515 gelesen werden.

Diese Ausgabe bezieht sich auf Version 7 Release 5 von IBM® WebSphere MQ und auf alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Wenn Sie Informationen an IBMsenden, erteilen Sie IBM ein nicht ausschließliches Recht, die Informationen in beliebiger Weise zu verwenden oder zu verteilen, ohne dass eine Verpflichtung für Sie entsteht.

Inhaltsverzeichnis

Referenzinformationen zum Entwickeln von Anwendungen

Die Informationen in diesem Abschnitt helfen Ihnen bei der Entwicklung Ihrer IBM WebSphere MQ-Anwendungen:

Zugehörige Tasks

Anwendungen entwickeln

Referenzinformationen zu MQI-Anwendungen

Verwenden Sie die Informationen der Links in diesem Kapitel beim Entwickeln Ihrer MQI-Anwendungen:

- „Codebeispiele” auf Seite 8
- „Konstanten” auf Seite 51
- „Im MQI verwendete Datentypen” auf Seite 217
- „Funktionsaufrufe” auf Seite 616
- „Attribute von Objekten” auf Seite 797
- „Rückgabecodes” auf Seite 873
- „Regeln zur Überprüfung von MQI-Optionen” auf Seite 874
- „Maschinencodierungen” auf Seite 901
- „Report options and message flags” auf Seite 904
- „Datenkonvertierungsexit” auf Seite 908
- „Als MQRFH2-Elemente angegebene Eigenschaften” auf Seite 932
- „Codepagekonvertierung” auf Seite 941

Zugehörige Konzepte

„Referenzinformationen zu Benutzerexits, API-Exits und installierbaren Services” auf Seite 1035
Verwenden Sie die in diesem Abschnitt bereitgestellten Links als Unterstützung bei der Entwicklung Ihrer Anwendungen für Benutzerexits, API-Exits und installierbare Services:

„IBM WebSphere MQ -Klassen für Java-Bibliotheken” auf Seite 1458
Die Position der IBM WebSphere MQ -Klassen für Java-Bibliotheken variiert je nach Plattform. Geben Sie diesen Standort an, wenn Sie eine Anwendung starten.

Zugehörige Tasks

Anwendungen entwickeln

Zugehörige Verweise

„SOAP-Referenz” auf Seite 976
Die Referenzinformationen zu WebSphere MQ Transport for SOAP sind alphabetisch angeordnet.

„Referenzmaterial für IBM WebSphere MQ Bridge for HTTP” auf Seite 1242
Referenzthemen für IBM WebSphere MQ Bridge for HTTP in alphabetischer Reihenfolge

„IBM WebSphere MQ .NET-Klassen und -Schnittstellen” auf Seite 1278
Die .NET-Klassen und -Schnittstellen von IBM WebSphere MQ sind alphabetisch aufgelistet. Die Eigenschaften, Methoden und Konstruktoren werden beschrieben.

„IBM WebSphere MQ-C++-Klassen” auf Seite 1343
Die IBM WebSphere MQ-C++-Klassen binden die IBM WebSphere MQ Message Queue Interface (MQI) mit ein. Die einzelne C++-Headerdatei **imqi.hpp** deckt all diese Klassen ab.

Zugehörige Informationen

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

Codebeispiele

Führen Sie mithilfe der Referenzinformationen in diesem Abschnitt die Tasks aus, die Ihrer Bedarfssituation entsprechen.

Beispiele für Programmiersprache C

Diese Themensammlung wurde überwiegend aus den WebSphere MQ for z/OS-Musteranwendungen übernommen. Sie gelten für alle Plattformen, sofern nicht anders angegeben.

Verbindung mit einem Warteschlangenmanager herstellen

Dieses Beispiel veranschaulicht, wie mit einem MQCONN-Aufruf in z/OS-Stapel eine Verbindung mit einem Programm hergestellt wird.

Dieser Auszug wurde aus der Beispielanwendung für die Suche (Programm CSQ4BCA1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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;
      }
.
   }
```

Verbindung von einem Warteschlangenmanager trennen

Dieses Beispiel veranschaulicht, wie mit dem MQDISC-Aufruf die Verbindung eines Warteschlangenmanagers in z/OS-Stapel getrennt wird.

In diesem Codeauszug werden die Variablen verwendet, die im Abschnitt „Verbindung mit einem Warteschlangenmanager herstellen” auf Seite 8 festgelegt wurden. Dieser Auszug wurde aus der Beispielanwendung für die Suche (Programm CSQ4BCA1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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;
      }
⋮
```

Erstellen einer dynamischen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf zum Erstellen einer dynamischen Warteschlange verwendet wird.

Dieser Auszug wurde aus einer Beispielanwendung für Nachrichtenverwaltung (Programm CSQ4TCD1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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;
}
.
```

Öffnen einer vorhandenen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf verwendet wird, um eine bereits definierte Warteschlange zu öffnen.

Dieser Auszug wurde aus der Beispielanwendung für die Suche (Programm CSQ4BCA1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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 */
```

Schließen einer Warteschlange

Dieses Beispiel veranschaulicht, wie der MQCLOSE-Aufruf verwendet wird, um eine Warteschlange zu schließen.

Dieser Auszug wurde aus der Beispielanwendung für die Suche (Programm CSQ4BCA1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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;
      }
⋮
```

Einreihen einer Nachricht mithilfe von MQPUT

Dieses Beispiel veranschaulicht, wie der MQPUT-Aufruf verwendet wird, um eine Nachricht in eine Warteschlange zu stellen.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie in den Abschnitten Beispielprogramme (Plattformen mit Ausnahme von 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 */
        }
}
```

Einreihen einer Nachricht mithilfe von MQPUT1

Dieses Beispiel veranschaulicht, wie der Aufruf von MQPUT1 verwendet wird, um eine Warteschlange zu öffnen, eine einzelne Nachricht in die Warteschlange zu stellen und dann die Warteschlange zu schließen.

Dieser Auszug wurde aus der Beispielanwendung für eine Prüfung der Kreditwürdigkeit (Programm CSQ4CCB5) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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;
   }
.
.
.
```

Abrufen einer Nachricht

Dieses Beispiel veranschaulicht, wie der MQGET-Aufruf verwendet wird, um eine Nachricht aus einer Warteschlange zu entfernen.

Dieser Auszug wurde aus der Beispielanwendung für die Suche (Programm CSQ4BCA1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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 */
```

Abrufen einer Nachricht mithilfe der Option für Warten

Dieses Beispiel veranschaulicht, wie Sie die Option "wait" des MQGET-Aufrufs verwenden.

Dieser Code akzeptiert abgeschnittene Nachrichten. Dieser Auszug wurde aus der Beispielanwendung für eine Prüfung der Kreditwürdigkeit (Programm CSQ4CCB5) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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();
         }
.
```

Abrufen einer Nachricht mithilfe von Signalisierung

Die Signalübertragung ist nur bei WebSphere MQ for z/OS verfügbar.

Dieses Beispiel veranschaulicht, wie Sie mit dem Aufruf MQGET ein Signal festlegen können, damit Sie benachrichtigt werden, sobald eine geeignete Nachricht in der Warteschlange eintrifft. Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
.
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;
}
```

Abfragen der Attribute eines Objekts

Dieses Beispiel veranschaulicht, wie der MQINQ-Aufruf verwendet wird, um die Attribute einer Warteschlange zu untersuchen.

Dieser Auszug wurde aus der Beispielanwendung für Warteschlangenattribute (Programm CSQ4CCC1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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 */
```

Attribute einer Warteschlange festlegen

Dieses Beispiel veranschaulicht, wie der MQSET-Aufruf verwendet wird, um die Attribute einer Warteschlange zu ändern.

Dieser Auszug wurde aus der Beispielanwendung für Warteschlangenattribute (Programm CSQ4CCC1) übernommen, die mit WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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 */
```

Statusinformationen mit MQSTAT abrufen

Dieses Beispiel veranschaulicht, wie Sie einen asynchronen MQPUT-Aufruf ausgeben und die Statusinformationen mit MQSTAT abrufen.

Dieser Auszug stammt aus der Beispielanwendung "Calling MQSTAT" (Programm amqsapt0), die im Rahmen von WebSphere MQ for Windows-Systeme bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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);
}
```

COBOL-Beispiele

Diese Themensammlung wurde aus den WebSphere MQ for z/OS-Musteranwendungen übernommen. Sie sind auf allen Plattformen anwendbar, sofern nicht ausdrücklich anders erwähnt.

Verbindung mit einem Warteschlangenmanager herstellen

Dieses Beispiel veranschaulicht, wie mit einem MQCONN-Aufruf in z/OS-Stapel eine Verbindung mit einem Programm hergestellt wird.

Dieser Auszug stammt aus der Beispielanwendung "Browse" (Programm CSQ4BVA1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
⋮
```

Verbindung von einem Warteschlangenmanager trennen

Dieses Beispiel veranschaulicht, wie mit dem MQDISC-Aufruf die Verbindung eines Warteschlangenmanagers in z/OS-Stapel getrennt wird.

In diesem Codeauszug werden die Variablen verwendet, die im Abschnitt „Verbindung mit einem Warteschlangenmanager herstellen” auf Seite 24 festgelegt wurden. Dieser Auszug stammt aus der Beispielanwendung "Browse" (Programm CSQ4BVA1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt

wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
⋮
```

Erstellen einer dynamischen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf zum Erstellen einer dynamischen Warteschlange verwendet wird.

Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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
*
```

Öffnen einer vorhandenen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf verwendet wird, um eine vorhandene Warteschlange zu öffnen.

Dieser Auszug stammt aus der Beispielanwendung "Browse" (Programm CSQ4BVA1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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
```

Schließen einer Warteschlange

Dieses Beispiel veranschaulicht, wie der MQCLOSE-Aufruf verwendet wird.

In diesem Codeauszug werden die Variablen verwendet, die im Abschnitt „Verbindung mit einem Warteschlangenmanager herstellen” auf Seite 24 festgelegt wurden. Dieser Auszug stammt aus der Beispielanwendung "Browse" (Programm CSQ4BVA1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt

wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
*
```

Einreihen einer Nachricht mithilfe von MQPUT

Dieses Beispiel veranschaulicht, wie der MQPUT-Aufruf mit dem Kontext verwendet wird.

Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
```

Einreihen einer Nachricht mithilfe von MQPUT1

Dieses Beispiel veranschaulicht, wie der Aufruf von MQPUT1 verwendet wird.

Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB5), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
*
```

Abrufen einer Nachricht

Dieses Beispiel veranschaulicht, wie der MQGET-Aufruf verwendet wird, um eine Nachricht aus einer Warteschlange zu entfernen.

Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
```

Abrufen einer Nachricht mithilfe der Option für Warten

Dieses Beispiel veranschaulicht, wie der MQGET-Aufruf mit der Option 'wait' verwendet und abgeschnittene Nachrichten akzeptiert werden.

Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB5), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
⋮
```

Abrufen einer Nachricht mithilfe von Signalisierung

Dieses Beispiel veranschaulicht die Verwendung des Aufrufs MQGET mit einer Signalübertragung. Dieser Auszug stammt aus der Beispielanwendung "Credit Check" (Programm CSQ4CVB2), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird.

Die Signalübertragung ist nur bei WebSphere MQ for z/OS verfügbar.

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

Abfragen der Attribute eines Objekts

Dieses Beispiel veranschaulicht, wie der MQINQ-Aufruf verwendet wird, um die Attribute einer Warteschlange zu untersuchen.

Dieser Auszug stammt aus der Beispielanwendung "Queue Attributes" (Programm CSQ4CVC1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Positionen der Beispielanwendungen auf anderen Plattformen finden Sie unter Beispielprogramme (Plattformen mit Ausnahme von 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.
⋮
```

Attribute einer Warteschlange festlegen

Dieses Beispiel veranschaulicht, wie die Attribute einer Warteschlange mit dem Aufruf MQSET geändert werden können.

Dieser Auszug stammt aus der Beispielanwendung "Queue Attributes" (Programm CSQ4CVC1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird. Die Namen und Speicherpositionen der Beispielanwendung auf anderen Plattformen finden Sie unter Beispielanwendungen (Plattformen mit Ausnahme von 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.
```

System/390-Assemblerbeispiele

Diese Themensammlung wurde überwiegend aus den WebSphere MQ for z/OS-Musteranwendungen übernommen.

Verbindung mit einem Warteschlangenmanager herstellen

Dieses Beispiel veranschaulicht, wie mit einem MQCONN-Aufruf in z/OS-Stapel eine Verbindung mit einem Programm hergestellt wird.

Dieser Auszug stammt aus dem Beispielprogramm "Browse" (CSQ4BAA1), das im Rahmen von WebSphere MQ for z/OS bereitgestellt wird.

```
⋮
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
*
```

Verbindung von einem Warteschlangenmanager trennen

Dieses Beispiel veranschaulicht, wie mit dem MQDISC-Aufruf die Verbindung eines Warteschlangenmanagers in z/OS-Stapel getrennt wird.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*
*        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
```

Erstellen einer dynamischen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf zum Erstellen einer dynamischen Warteschlange verwendet wird.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*
*    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
```

Öffnen einer vorhandenen Warteschlange

Dieses Beispiel veranschaulicht, wie der MQOPEN-Aufruf verwendet wird, um eine bereits definierte Warteschlange zu öffnen.

Dieses Beispiel veranschaulicht die Angabe von zwei Optionen. Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*
*    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
```

Schließen einer Warteschlange

Dieses Beispiel veranschaulicht, wie der MQCLOSE-Aufruf verwendet wird, um eine Warteschlange zu schließen.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
```

```
.
*
* 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
```

Einreihen einer Nachricht mithilfe von MQPUT

Dieses Beispiel veranschaulicht, wie der MQPUT-Aufruf verwendet wird, um eine Nachricht in eine Warteschlange zu stellen.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
.
*     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
```

Einreihen einer Nachricht mithilfe von MQPUT1

Dieses Beispiel veranschaulicht, wie der Aufruf von MQPUT1 verwendet wird, um eine Warteschlange zu öffnen, eine einzelne Nachricht in die Warteschlange zu stellen und dann die Warteschlange zu schließen.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*
```

```
*     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
```

Abrufen einer Nachricht

Dieses Beispiel veranschaulicht, wie der MQGET-Aufruf verwendet wird, um eine Nachricht aus einer Warteschlange zu entfernen.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*
*     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
```

Abrufen einer Nachricht mithilfe der Option für Warten

Dieses Beispiel veranschaulicht, wie Sie die Option "wait" des MQGET-Aufrufs verwenden.

Dieser Code akzeptiert abgeschnittene Nachrichten. Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```
⋮
*     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
```

Abrufen einer Nachricht mithilfe von Signalisierung

Dieses Beispiel veranschaulicht, wie der MQGET-Aufruf verwendet wird, um ein Signal so zu setzen, dass Sie benachrichtigt werden, wenn eine geeignete Nachricht in einer Warteschlange eintrifft.

Dieser Auszug stammt nicht aus den Beispielanwendungen, die im Rahmen von WebSphere MQ bereitgestellt werden.

```

```

```
⋮
*
*     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
```

Attribute einer Warteschlange abfragen und festlegen

Dieses Beispiel veranschaulicht, wie der MQINQ-Aufruf zum Abfragen der Attribute einer Warteschlange und zum Verwenden des MQSET-Aufrufs verwendet wird, um die Attribute einer Warteschlange zu ändern.

Dieser Auszug stammt aus der Beispielanwendung "Queue Attributes" (Programm CSQ4CAC1), die im Rahmen von WebSphere MQ for z/OS bereitgestellt wird.

```
:
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?
⋮
```

Konstanten

Führen Sie mithilfe der Referenzinformationen in diesem Abschnitt die Tasks aus, die Ihrer Bedarfssituation entsprechen.

IBM WebSphere MQ-Kopier-, Header-, Include- und Moduldateien

Bei den Informationen in diesem Abschnitt handelt es sich um Informationen zur allgemeinen Programmierschnittstelle.

In diesem Abschnitt finden Sie Informationen zur Verwendung des Message Queue Interface für verschiedene Programmiersprachen.

C-Headerdateien

Headerdateien erleichtern Ihnen das Schreiben von C-Anwendungsprogrammen, die MQI verwenden. Diese Headerdateien werden in der Tabelle zusammengefasst:

Tabelle 1. C-Headerdateien — Aufrufprototypen, Datentypen, Rückkehrcodes, Konstanten und Strukturen

Dateiname	Beschreibung	IBM i	UNIX -und Linux® -Systeme	Windows	z/OS
Aufrufprototypen, Datentypen, Rückkehrcodes, Konstanten und Strukturen					
CMQC	MQI-Definitionen	C	C	C	C
CMQBC	MQAI-Definitionen	C	C	C	
CMQEC	Definition Schnittstelleneingangspunkte (einschließlich CMQC, CMQXC und CMQZC)		C	C	
CMQCFC	PCF-Definitionen	C	C	C	C
CMQPSC	Publish/Subscribe-Definitionen	C	C	C	C

Tabelle 1. C-Headerdateien — Aufrufprototypen, Datentypen, Rückkehrcodes, Konstanten und Strukturen (Forts.)					
Dateiname	**Beschreibung**	**IBM i**	**UNIX -und Linux® -Systeme**	**Windows**	**z/OS**
CMQXC	Kanal-Exit-Definitionen	C	C	C	C
CMQZC	Definitionen für installierbare Services	C	C	C	
Schlüssel: C= Bereitgestellte Dateien					

COPY-Dateien für COBOL

Die verschiedenen COPY-Dateien erleichtern Ihnen das Schreiben von Anwendungsprogrammen für COBOL, die MQI verwenden. Diese Dateien werden in der Tabelle zusammengefasst:

Tabelle 2. COBOL-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen					
Dateiname	**Beschreibung**	**IBM i**	**UNIX-Systeme**	**Windows**	**z/OS**
Rückkehrcodes und Konstanten					
CMQx	MQI-Definitionen	V	V	V	V
CMQCFx	PCF-Definitionen	V	V	V	V
CMQPSx	Publish/Subscribe-Definitionen	V	V	V	V
CMQXx	Kanal-Exit-Definitionen	V	V	V	V
Strukturen					
CMQAIRx	MQAIR - Datensätze für Authentifizierungsinformationen		V L	V L	
CMQBOx	MQBO - Startoptionen	V L	V L	V L	
CMQCDx	MQCD - Kanaldefinition	V L	V L	V L	V L
CMQCFBFx	MQCFBF - PCF-Parameter Bytefolgefilter	V L	V L	V L	V L
CMQCFBSx	MQCFBS - PCF-Parameter Bytefolge	V L	V L	V L	V L
CMQCFGRx	MQCFGR - PCF-Gruppenparameter	V L	V L	V L	V L
CMQCFHx	MQCFH - PCF-Header	V L	V L	V L	V L
CMQCFIFx	MQCFIF - PCF-Parameter Integer-Filter	V L	V L	V L	V L
CMQCFILx	MQCFIL - PCF-Parameter Integer-Liste	V L	V L	V L	V L
CMQCFINx	MQCFIN - PCF-Parameter Integer	V L	V L	V L	V L
CMQCFSFx	MQCFSF - PCF-Parameter Zeichenfolgefilter	V L	V L	V L	V L
CMQCFSLx	MQCFSL - PCF-Zeichenfolgenlistenparameter	V L	V L	V L	V L
CMQCFSTx	MQCFST - PCF-Zeichenfolgeparameter	V L	V L	V L	V L

Tabelle 2. COBOL-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen (Forts.)

Dateiname	Beschreibung	IBM i	UNIX-Systeme	Windows	z/OS
CMQCFXLx	MQCFIL64 - PCF-Parameter 64-Bit-Integer-Liste	V L	V L	V L	V L
CMQCFXNx	MQCFIN64 - PCF-Parameter 64-Bit-Integer	V L	V L	V L	V L
CMQCHRVx	MQCHARV - Zeichenfolge variabler Länge	V L	V L	V L	V L
CMQCIHx	MQCIH - Header für CICS-Bridge	V L	V L	V L	V L
CMQCNOx	MQCNO - Verbindungsoptionen	V L	V L	V L	V L
CMQCSPx	MQCSP - Sicherheitsparameter	V L	V L	V L	V L
CMQCXPx	MQCXP - Kanal-Exit-Parameter	V L			V L
CMQDHx	MQDH - Verteilerheader	V L	V L	V L	V L
CMQDLHx	MQDLH - Header für nicht zustellbare Nachrichten	V L	V L	V L	V L
CMQDXPx	MQDXP - Exit-Parameter für Datenkonvertierung	V L		V L	
CMQEPHx	MQEPH - Eingebetteter PCF-Header	V L	V L	V L	V L
CMQGMOx	MQGMO - Optionen zum Abrufen von Nachrichten	V L	V L	V L	V L
CMQIIHx	MQIIH - IMS-Header	V L	V L	V L	V L
CMQMDx	MQMD - Nachrichtendeskriptor	V L	V L	V L	V L
CMQMD1x	MQMD1 - Nachrichtendeskriptor Version 1	V L	V L	V L	V L
CMQMD2x	MQMD2 - Nachrichtendeskriptor Version 2	V L	V L	V L	V L
CMQMDEx	MQMDE - Erweiterter Nachrichtendeskriptor	V L	V L	V L	V L
CMQODx	MQOD - Objektdeskriptor	V L	V L	V L	V L
CMQORx	MQOR - Objektdatensatz	V L	V L	V L	V L
CMQPMOx	MQPMO - Optionen zum Einreihen von Nachrichten	V L	V L	V L	V L
CMQRFHx	MQRFH - Header für Regeln und Formatierung	V L	V L	V L	V L
CMQRFH2x	MQRFH2 - Header 2 für Regeln und Formatierung	V L	V L	V L	V L
CMQRMHx	MQRMH - Header für Referenznachrichten	V L	V L	V L	V L
CMQRRx	MQRR - Antwortdatensatz	V L	V L	V L	
CMQSCOx	MQSCO - SSL-Konfigurationsoptionen		V L	V L	

Tabelle 2. COBOL-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen (Forts.)					
Dateiname	**Beschreibung**	**IBM i**	**UNIX-Systeme**	**Windows**	**z/OS**
CMQTMx	MQTM - Auslösenachricht	V L		V L	V L
CMQTMCx	MQTMC - Zeichen für Auslösenachricht	V L	V L		
CMQTMC2x	MQTMC2 - Zeichen für Auslösenachricht 2	V L	V L	V L	V L
CMQWIHx	MQWIH - Auslastungs-Header	V L	V L	V L	V L
CMQXQHx	MQXQH - Header für die Übertragungswarteschlange	V L	V L	V L	V L
Schlüssel: • Bereitgestellte Dateien mit Anfangswerten, x=V • Bereitgestellte Dateien ohne Anfangswerte, x=L					

PL/I-Kopfdatendateien

Die folgenden INCLUDE-Dateien werden für die Programmiersprache PL/I bereitgestellt. Diese Dateien sind nur unter z/OS verfügbar.

Tabelle 3. PL/I-Include-Dateien — Datentypen, Rückkehrcodes, Konstanten und Strukturen					
Dateiname	**Beschreibung**	**IBM i**	**UNIX-Systeme**	**Windows**	**z/OS**
Datentypen, Rückkehrcodes, Konstanten und Strukturen					
CMQP	MQI-Definitionen				P
CMQCFP	PCF-Definitionen				P
CMQEPP	Definitionen für den Eingangspunkt				P
CMQPSP	Publish/Subscribe-Definitionen				P
CMQXP	Kanal-Exit-Definitionen				P
Schlüssel: P= Bereitgestellte Datei					

COPY-Dateien für RPG

Die folgenden COPY-Dateien werden für die Programmiersprache RPG bereitgestellt. Diese Dateien sind nur in IBM i verfügbar.

Tabelle 4. RPG-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen					
Dateiname	**Beschreibung**	**IBM i**	**UNIX-Systeme**	**Windows**	**z/OS**
Rückkehrcodes und Konstanten					
CMQx	MQI-Definitionen	G R			
CMQCFx	PCF-Definitionen	G			

Tabelle 4. RPG-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen (Forts.)					
Dateiname	**Beschreibung**	**IBM i**	**UNIX-Systeme**	**Windows**	**z/OS**
CMQPSx	Publish/Subscribe-Definitionen	G			
CMQXx	Kanal-Exit-Definitionen	G R			
Strukturen					
CMQBOx	MQBO - Startoptionen	G H			
CMQCDx	MQCD - Kanaldefinition	G H R			
CMQCFBFx	MQCFBF - PCF-Parameter Bytefolgefilter	G H			
CMQCFBSx	MQCFBS - PCF-Parameter Bytefolge	G H			
CMQCFGRx	MQCFGR - PCF-Gruppenparameter	G H			
CMQCFHx	MQCFH - PCF-Header	G H			
CMQCFIFx	MQCFIF - PCF-Parameter Integer-Filter	G H			
CMQCFILx	MQCFIL - PCF-Parameter Integer-Liste	G H			
CMQCFINx	MQCFIN - PCF-Parameter Integer	G H			
CMQCFSFx	MQCFSF - PCF-Parameter Zeichenfolgefilter	G H			
CMQCFSLx	MQCFSL - PCF-Zeichenfolgenlistenparameter	G H			
CMQCFSTx	MQCFST - PCF-Zeichenfolgeparameter	G H			
CMQCFXLx	MQCFIL64 - PCF-Parameter 64-Bit-Integer-Liste	G H			
CMQCFXNx	MQCFIN64 - PCF-Parameter 64-Bit-Integer	G H			
CMQCHARVx	MQCHARV - Zeichenfolge variabler Länge	G H			
CMQCIHx	MQCIH - Header für CICS-Bridge	G H			
CMQCNOx	MQCNO - Verbindungsoptionen	G H			
CMQCSPx	MQCSP - Sicherheitsparameter	G H			
CMQCXPx	MQCXP - Kanal-Exit-Parameter	G H R			
CMQDHx	MQDH - Verteilerheader	G H R			
CMQDLHx	MQDLH - Header für nicht zustellbare Nachrichten	G H R			
CMQDXPx	MQDXP - Exit-Parameter für Datenkonvertierung	G H R			

Tabelle 4. RPG-Copy-Dateien - Rückkehrcodes, Konstanten und Strukturen (Forts.)

Dateiname	Beschreibung	IBM i	UNIX-Systeme	Windows	z/OS
CMQEPHx	MQEPH - Eingebetteter PCF-Header	G H			
CMQGMOx	MQGMO - Optionen zum Abrufen von Nachrichten	G H R			
CMQIIHx	MQIIH - IMS-Header	G H R			
CMQMDx	MQMD - Nachrichtendeskriptor	G H R			
CMQMD1x	MQMD1 - Nachrichtendeskriptor Version 1	G H R			
CMQMD2x	MQMD2 - Nachrichtendeskriptor Version 2	G H			
CMQMDEx	MQMDE - Erweiterter Nachrichtendeskriptor	G H R			
CMQODx	MQOD - Objektdeskriptor	G H R			
CMQORx	MQOR - Objektdatensatz	G H R			
CMQPMOx	MQPMO - Optionen zum Einreihen von Nachrichten	G H R			
CMQPXPx	MQPXP - Routingexitparameter für Publish/Subscribe	G H			
CMQRFHx	MQRFH - Header für Regeln und Formatierung	G H			
CMQRFH2x	MQRFH2 - Header 2 für Regeln und Formatierung	G H			
CMQRMHx	MQRMH - Header für Referenznachrichten	G H R			
CMQRRx	MQRR - Antwortdatensatz	G H R			
CMQTMx	MQTM - Auslösenachricht	G H R			
CMQTMCx	MQTMC - Zeichen für Auslösenachricht	G H R			
CMQTMC2x	MQTMC2 - Zeichen für Auslösenachricht 2	G H R			
CMQWIHx	MQWIH - Auslastungs-Header	G H			
CMQXQHx	MQXQH - Header für die Übertragungswarteschlange	G H R			

Schlüssel:

- Bereitgestellte Datei für statische Verbindung, initialisiert, x=G
- Bereitgestellte Datei für statische Verbindung, nicht initialisiert, x=H
- Bereitgestellte Datei für dynamische Verbindung, initialisiert, x=R

Visual Basic-Moduldateien

Header- oder Form-Dateien erleichtern Ihnen das Schreiben von Visual Basic-Anwendungsprogrammen, die MQI verwenden. Diese Headerdateien werden nur in 32-Bit-Versionen bereitgestellt und sind in dieser Tabelle zusammengefasst:

Tabelle 5. Visual Basic-Moduldateien — Aufrufdeklarationen, Datentypen, Rückkehrcodes, Konstanten und Strukturen

Dateiname	Beschreibung	IBM i	UNIX and Linux-Sys-teme	Windows	z/OS
Aufrufdeklarationen, Datentypen, Rückkehrcodes, Konstanten und Strukturen					
CMQB	MQI-Definitionen			B	
CMQBB	MQAI-Definitionen			B	
CMQCFB	PCF-Definitionen			B	
CMQXB	Kanal-Exit-Definitionen			B	
Schlüssel: B= Bereitgestellte Datei					

MQ_* (Zeichenfolgenlängen)

Tabelle 6. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'
MQ_APPL_NAME_LENGTH	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'
MQ_APPL_TAG_LENGTH	28	X'0000001C'
MQ_ARM_SUFFIX_LENGTH	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'

Tabelle 6. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
MQ_CLIENT_ID_LENGTH	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH	48	X'00000030'
MQ_CONN_NAME_LENGTH	264	X'00000108'
MQ_CONN_TAG_LENGTH	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_DISTINGUISHED_NAME_LENGTH	1024	X'00000400'
MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
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_LIKE_LENGTH	4	X'00000004'
MQ_FORMAT_LENGTH	8	X'00000008'
MQ_FUNCTION_LENGTH	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_JOB_NAME_LENGTH	28	X'0000001C'
MQ_MCA_NAME_LENGTH	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'

Tabelle 6. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
MQ_MSG_ID_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'
MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
MQ_PASSWORD_LENGTH	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'
MQ_PROCESS_NAME_LENGTH	48	X'00000030'
MQ_PROCESS_USER_DATA_LENGTH	128	X'00000080'
MQ_PROGRAM_NAME_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_COMMAND_LENGTH	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'

Tabelle 6. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQ_SUB_POINT_LENGTH	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_B_NONE	1	X'00000001'
MQ_SUITE_B_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_TOPIC_NAME_LENGTH	48	X'00000030'
MQ_TOPIC_STR_LENGTH	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	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_VERSION_LENGTH	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

MQ_* (Befehlsformat, Zeichenfolgelängen)

Tabelle 7. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
MQ_ARCHIVE_UNIT_LENGTH	8	X'00000008'
MQ_ASID_LENGTH	4	X'00000004'
MQ_AUTH_PROFILE_NAME_LENGTH	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH	32768	X'00008000'
MQ_DATA_SET_NAME_LENGTH	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
MQ_DSG_NAME_LENGTH	8	X'00000008'
MQ_ENTITY_NAME_LENGTH	64	X'00000040'
MQ_ENV_INFO_LENGTH	96	X'00000060'
MQ_IP_ADDRESS_LENGTH	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
MQ_LOG_PATH_LENGTH	1024	X'00000400'
MQ_LRSN_LENGTH	12	X'0000000C'
MQ_ORIGIN_NAME_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'
MQ_RBA_LENGTH	12	X'0000000C'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
MQ_UOW_ID_LENGTH	256	X'00000100'
MQ_USER_DATA_LENGTH	10240	X'00002800'
MQ_VOLSER_LENGTH	6	X'00000006'

MQACH_* (Headerstruktur API-Exit-Verkettungsbereich)

Tabelle 8. Strukturen von Konstanten

Ihren Namen	Struktur
MQACH_STRUC_ID	"ACH¬"
MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 9. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQACH_VERSION_1	1	X'00000001'
MQACH_CURRENT_VERSION	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	
MQACH_CURRENT_LENGTH	(value differs by platform or version)	

MQACT_* (Berechnungstoken)

Tabelle 10. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQACT_NONE	X'00...00' (32 Nullen)
MQACT_NONE_ARRAY	'\0','\0',... (32 Nullen)

MQACT_* (Befehlsformat, Aktionsoptionen)

Tabelle 11. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG	2	X'00000002'
MQACT_COLLECT_STATISTICS	3	X'00000003'
MQACT_PUBSUB	4	X'00000004'

MQACTP_* (Aktion)

Tabelle 12. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQACTP_NEW	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
MQACTP_REPORT	3	X'00000003'

MQACTT_* (Typen Berechnungstoken)

Tabelle 13. Werte von Konstanten

Ihren Namen	Hexadezimalwert
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'

Tabelle 13. Werte von Konstanten (Forts.)	
Ihren Namen	**Hexadezimalwert**
MQACTT_USER	X'19'

MQADOPT_* (Übernahme Neue MCA-Prüfungen und Übernahme Neue MCA-Typen)

Übernahme Neue MCA-Prüfungen

Tabelle 14. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Übernahme Neue MCA-Typen

Tabelle 15. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Datensatzstruktur Authentifizierungsinformationen)

Tabelle 16. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQAIR_STRUC_ID	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 17. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
MQAIR_CURRENT_VERSION	2	X'00000002'

MQAIT_* (Typ Authentifizierungsinformationen)

Tabelle 18. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAIT_ALL	0	X'00000000'

Tabelle 18. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'

MQAS_* (Befehlsformat asynchrone Statuswerte)

Tabelle 19. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAS_NONE	0	X'00000000'
MQAS_STARTED	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_STOPPED	3	X'00000003'
MQAS_SUSPENDED	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ACTIVE	6	X'00000006'
MQAS_INACTIVE	7	X'00000007'

MQAT_* (PUT-Anwendungstypen)

Tabelle 20. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAT_UNKNOWN	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'
MQAT_OS390	2	X'00000002'
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_WINDOWS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIAN	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'

Tabelle 20. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAT_IMS_BRIDGE	19	X'00000013'
MQAT_XCF	20	X'00000014'
MQAT_CICS_BRIDGE	21	X'00000015'
MQAT_NOTES_AGENT	22	X'00000016'
MQAT_TPF	23	X'00000017'
MQAT_USER	25	X'00000019'
MQAT_BROKER	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'
MQAT_BATCH	32	X'00000020'
MQAT_RRS_BATCH	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	
MQAT_USER_FIRST	65536	X'00010000'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_* (Befehlsformat, Berechtigungswerte)

Tabelle 21. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_CREATE	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'
MQAUTH_PASS_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT	16	X'00000010'

Tabelle 21. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAUTH_CONTROL	17	X'00000011'
MQAUTH_CONTROL_EXTENDED	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUBSCRIBE	20	X'00000014'
MQAUTH_RESUME	21	X'00000015'
MQAUTH_SYSTEM	22	X'00000016'

MQAUTHOPT_* (Befehlsformat Berechtigungsoptionen)

Tabelle 22. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAUTHOPT_CUMULATIVE	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'
MQAUTHOPT_ENTITY_SET	2	X'00000002'
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLICIT	16	X'00000010'

MQAXC_* (API-Exit-Kontextstruktur)

Tabelle 23. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQAXC_STRUC_ID	"AXC¬"
MQAXC_STRUC_ID_ARRAY	'A','X','C','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 24. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAXC_VERSION_1	1	X'00000001'
MQAXC_CURRENT_VERSION	1	X'00000001'

MQAXP_* (API-Exit-Parameterstruktur)

Tabelle 25. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQAXP_STRUC_ID	"AXP¬"
MQAXP_STRUC_ID_ARRAY	'A','X','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 26. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'

Tabelle 26. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQAXP_CURRENT_VERSION	2	X'00000002'

MQBA_* (Byteattribut-Selektoren)

Tabelle 27. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQBA_FIRST	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_* (Befehlsformat Byteparametertypen)

Tabelle 28. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQBACF_FIRST	7001	X'00001B59'
MQBACF_EVENT_ACCOUNTING_TOKEN	7001	X'00001B59'
MQBACF_EVENT_SECURITY_ID	7002	X'00001B5A'
MQBACF_RESPONSE_SET	7003	X'00001B5B'
MQBACF_RESPONSE_ID	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_* (Pufferlänge für mqAddString und mqSetString)

Tabelle 29. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQBL_NULL_TERMINATED	-1	X'FFFFFFFF'

MQBMHO_* (Optionen und Struktur Puffer an Nachrichtenkennung)

Struktur Optionen Puffer an Nachrichtenkennung

Tabelle 30. Strukturen von Konstanten

Ihren Namen	Struktur
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B','M','H','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 31. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

Optionen Puffer an Nachrichtenkennung

Tabelle 32. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIES	1	X'00000001'

MQBND_* (Standardbindungen)

Tabelle 33. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_* (Startoptionen und Struktur)

Struktur Startoptionen

Tabelle 34. Strukturen von Konstanten

Ihren Namen	Struktur
MQBO_STRUC_ID	"BO¬¬"
MQBO_STRUC_ID_ARRAY	'B','O','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 35. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

Startoptionen

Tabelle 36. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQBO_NONE	0	X'00000000'

MQBT_* (Befehlsformat Bridge-Typen)

Tabelle 37. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQBT_OTMA	1	X'00000001'

MQCA_* (Zeichenattribut-Selektoren)

Tabelle 38. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'
MQCA_ALTERATION_TIME	2028	X'000007EC'
MQCA_APPL_ID	2001	X'000007D1'
MQCA_AUTH_INFO_CONN_NAME	2053	X'00000805'
MQCA_AUTH_INFO_DESC	2046	X'000007FE'
MQCA_AUTH_INFO_NAME	2045	X'000007FD'
MQCA_AUTH_INFO_OCSP_URL	2109	X'0000083D'
MQCA_AUTO_REORG_CATALOG	2091	X'0000082B'
MQCA_AUTO_REORG_START_TIME	2090	X'0000082A'
MQCA_BACKOUT_REQ_Q_NAME	2019	X'000007E3'
MQCA_BASE_OBJECT_NAME	2002	X'000007D2'
MQCA_BASE_Q_NAME	2002	X'000007D2'
MQCA_BATCH_INTERFACE_ID	2068	X'00000814'
MQCA_CF_STRUC_DESC	2052	X'00000804'
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'

Tabelle 38. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCA_COMMAND_INPUT_Q_NAME	2003	X'000007D3'
MQCA_COMMAND_REPLY_Q_NAME	2067	X'00000813'
MQCA_CREATION_DATE	2004	X'000007D4'
MQCA_CREATION_TIME	2005	X'000007D5'
MQCA_DEAD_LETTER_Q_NAME	2006	X'000007D6'
MQCA_DEF_XMIT_Q_NAME	2025	X'000007E9'
MQCA_DNS_GROUP	2071	X'00000817'
MQCA_ENV_DATA	2007	X'000007D7'
MQCA_FIRST	2001	X'000007D1'
MQCA_IGQ_USER_ID	2041	X'000007F9'
MQCA_INITIATION_Q_NAME	2008	X'000007D8'
MQCA_LAST	4000	X'00000FA0'
MQCA_LAST_USED	2109	X'0000083D'
MQCA_LDAP_PASSWORD	2048	X'00000800'
MQCA_LDAP_USER_NAME	2047	X'000007FF'
MQCA_LU_GROUP_NAME	2072	X'00000818'
MQCA_LU_NAME	2073	X'00000819'
MQCA_LU62_ARM_SUFFIX	2074	X'0000081A'
MQCA_MODEL_DURABLE_Q	2096	X'00000830'
MQCA_MODEL_NON_DURABLE_Q	2097	X'00000831'
MQCA_MONITOR_Q_NAME	2066	X'00000812'
MQCA_NAMELIST_DESC	2009	X'000007D9'
MQCA_NAMELIST_NAME	2010	X'000007DA'
MQCA_NAMES	2020	X'000007E4'
MQCA_PARENT	2102	X'00000836'
MQCA_PASS_TICKET_APPL	2086	X'00000826'
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'

Tabelle 38. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCA_RESUME_TIME	2099	X'00000833'
MQCA_SERVICE_DESC	2078	X'0000081E'
MQCA_SERVICE_NAME	2077	X'0000081D'
MQCA_SERVICE_START_ARGS	2080	X'00000820'
MQCA_SERVICE_START_COMMAND	2079	X'0000081F'
MQCA_SERVICE_STOP_ARGS	2082	X'00000822'
MQCA_SERVICE_STOP_COMMAND	2081	X'00000821'
MQCA_STDERR_DESTINATION	2084	X'00000824'
MQCA_STDOUT_DESTINATION	2083	X'00000823'
MQCA_SSL_CRL_NAMELIST	2050	X'00000802'
MQCA_SSL_CRYPTO_HARDWARE	2051	X'00000803'
MQCA_SSL_KEY_LIBRARY	2069	X'00000815'
MQCA_SSL_KEY_MEMBER	2070	X'00000816'
MQCA_SSL_KEY_REPOSITORY	2049	X'00000801'
MQCA_STORAGE_CLASS	2022	X'000007E6'
MQCA_STORAGE_CLASS_DESC	2042	X'000007FA'
MQCA_SYSTEM_LOG_Q_NAME	2065	X'00000811'
MQCA_TCP_NAME	2075	X'0000081B'
MQCA_TOPIC_DESC	2093	X'0000082D'
MQCA_TOPIC_NAME	2092	X'0000082C'
MQCA_TOPIC_STRING_FILTER	2108	X'0000083C'
MQCA_TOPIC_STRING	2094	X'0000082E'
MQCA_TPIPE_NAME	2085	X'00000825'
MQCA_TRIGGER_CHANNEL_NAME	2064	X'00000810'
MQCA_TRIGGER_DATA	2023	X'000007E7'
MQCA_TRIGGER_PROGRAM_NAME	2062	X'0000080E'
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_* (Befehlsformat Zeichenparametertypen)

Tabelle 39. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCACF_FIRST	3001	X'00000BB9'

Tabelle 39. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_FROM_PROCESS_NAME	3003	X'00000BBB'
MQCACF_TO_PROCESS_NAME	3004	X'00000BBC'
MQCACF_FROM_NAMELIST_NAME	3005	X'00000BBD'
MQCACF_TO_NAMELIST_NAME	3006	X'00000BBE'
MQCACF_FROM_CHANNEL_NAME	3007	X'00000BBF'
MQCACF_TO_CHANNEL_NAME	3008	X'00000BC0'
MQCACF_FROM_AUTH_INFO_NAME	3009	X'00000BC1'
MQCACF_TO_AUTH_INFO_NAME	3010	X'00000BC2'
MQCACF_Q_NAMES	3011	X'00000BC3'
MQCACF_PROCESS_NAMES	3012	X'00000BC4'
MQCACF_NAMELIST_NAMES	3013	X'00000BC5'
MQCACF_ESCAPE_TEXT	3014	X'00000BC6'
MQCACF_LOCAL_Q_NAMES	3015	X'00000BC7'
MQCACF_MODEL_Q_NAMES	3016	X'00000BC8'
MQCACF_ALIAS_Q_NAMES	3017	X'00000BC9'
MQCACF_REMOTE_Q_NAMES	3018	X'00000BCA'
MQCACF_SENDER_CHANNEL_NAMES	3019	X'00000BCB'
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_CORREL_ID	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'

Tabelle 39. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCACF_CHILD_Q_MGR_NAME	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NAME	3043	X'00000BE3'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENT_USER_ID	3045	X'00000BE5'
MQCACF_OBJECT_NAME	3046	X'00000BE6'
MQCACF_EVENT_Q_MGR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NAMES	3048	X'00000BE8'
MQCACF_EVENT_APPL_IDENTITY	3049	X'00000BE9'
MQCACF_EVENT_APPL_NAME	3050	X'00000BEA'
MQCACF_EVENT_APPL_ORIGIN	3051	X'00000BEB'
MQCACF_SUBSCRIPTION_NAME	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY	3054	X'00000BEE'
MQCACF_REG_SUB_IDENTITY	3055	X'00000BEF'
MQCACF_SUBSCRIPTION_USER_DATA	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
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_ENTITY_NAME	3068	X'00000BFC'
MQCACF_SERVICE_COMPONENT	3069	X'00000BFD'
MQCACF_RESPONSE_Q_MGR_NAME	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_RESTART_LOG_EXTENT_NAME	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
MQCACF_LOG_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN	3079	X'00000C07'

Tabelle 39. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NAME	3082	X'00000C0A'
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'
MQCACF_CONFIGURATION_DATE	3091	X'00000C13'
MQCACF_CONFIGURATION_TIME	3092	X'00000C14'
MQCACF_FROM_CF_STRUC_NAME	3093	X'00000C15'
MQCACF_TO_CF_STRUC_NAME	3094	X'00000C16'
MQCACF_CF_STRUC_NAMES	3095	X'00000C17'
MQCACF_FAIL_DATE	3096	X'00000C18'
MQCACF_FAIL_TIME	3097	X'00000C19'
MQCACF_BACKUP_DATE	3098	X'00000C1A'
MQCACF_BACKUP_TIME	3099	X'00000C1B'
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'

Tabelle 39. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_OPERATION_DATE	3132	X'00000C3C'
MQCACF_OPERATION_TIME	3133	X'00000C3D'
MQCACF_ACTIVITY_DESC	3134	X'00000C3E'
MQCACF_APPL_IDENTITY_DATA	3135	X'00000C3F'
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
MQCACF_STRUC_ID	3142	X'00000C46'
MQCACF_VALUE_NAME	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'
MQCACF_DESTINATION	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'

Tabelle 39. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCACF_FROM_SUB_NAME	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
MQCACF_LAST_MSG_TIME	3167	X'00000C5F'
MQCACF_LAST_MSG_DATE	3168	X'00000C60'
MQCACF_SUBSCRIPTION_POINT	3169	X'00000C61'
MQCACF_FILTER	3170	X'00000C62'
MQCACF_NONE	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_LAST_USED	3172	X'00000C64'

MQCACH_* (Befehlsformat Zeichenkanalparametertypen)

Tabelle 40. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_USER_ID	3517	X'00000DBD'
MQCACH_PASSWORD	3518	X'00000DBE'
MQCACH_LOCAL_ADDRESS	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'

Tabelle 40. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCACH_MCA_JOB_NAME	3530	X'00000DCA'
MQCACH_LAST_LUWID	3531	X'00000DCB'
MQCACH_CURRENT_LUWID	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
MQCACH_MR_EXIT_USER_DATA	3535	X'00000DCF'
MQCACH_SSL_CIPHER_SPEC	3544	X'00000DD8'
MQCACH_SSL_PEER_NAME	3545	X'00000DD9'
MQCACH_SSL_HANDSHAKE_STAGE	3546	X'00000DDA'
MQCACH_SSL_SHORT_PEER_NAME	3547	X'00000DDB'
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'
MQCACH_SSL_CERT_USER_ID	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NAME	3551	X'00000DDF'
MQCACH_IP_ADDRESS	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_* (CICS-Header ADS-Deskriptoren)

Tabelle 41. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCADSD_NONE	0	X'00000000'
MQCADSD_SEND	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_* (Werte Verbindungsaffinität)

Tabelle 42. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFERRED	1	X'00000001'

MQCAMO_* (Befehlsformat, Zeichenüberwachungsparametertypen)

Tabelle 43. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCAMO_FIRST	2701	X'00000A8D'
MQCAMO_CLOSE_DATE	2701	X'00000A8D'
MQCAMO_CLOSE_TIME	2702	X'00000A8E'
MQCAMO_CONN_DATE	2703	X'00000A8F'
MQCAMO_CONN_TIME	2704	X'00000A90'
MQCAMO_DISC_DATE	2705	X'00000A91'
MQCAMO_DISC_TIME	2706	X'00000A92'
MQCAMO_END_DATE	2707	X'00000A93'
MQCAMO_END_TIME	2708	X'00000A94'
MQCAMO_OPEN_DATE	2709	X'00000A95'
MQCAMO_OPEN_TIME	2710	X'00000A96'
MQCAMO_START_DATE	2711	X'00000A97'
MQCAMO_START_TIME	2712	X'00000A98'
MQCAMO_LAST_USED	2712	X'00000A98'

MQCBC_* (Struktur MQCBC-Konstanten)

Tabelle 44. Strukturen von Konstanten

Ihren Namen	Struktur
MQCBC_STRUC_ID	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 45. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBCF_* (Markierungen MQCBC-Konstanten)

Tabelle 46. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCBCF_NONE	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_* (Typ 'Callback' MQCBC-Konstanten)

Tabelle 47. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
MQCBCT_REGISTER_CALL	3	X'00000003'

Tabelle 47. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Struktur MQCBD-Konstanten)

Tabelle 48. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQCBD_STRUC_ID	"CBD¬"
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 49. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCBD_VERSION_1	1	X'00000001'
MQCBD_CURRENT_VERSION	1	X'00000001'

MQCBDO_* ('Callback'-Optionen MQCBD-Konstanten)

Tabelle 50. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Erstellungsoptionen für mqCreateBag)

Tabelle 51. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_INHIBITED	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_DO_NOT_REORDER	0	X'00000000'

Tabelle 51. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCBO_CHECK_SELECTORS	8	X'00000008'
MQCBO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_* (MQCBD-Konstanten, Art der Callback-Funktion)

Tabelle 52. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_* (Beendigungscodes)

Tabelle 53. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCC_OK	0	X'00000000'
MQCC_WARNING	1	X'00000001'
MQCC_FAILED	2	X'00000002'
MQCC_UNKNOWN	-1	X'FFFFFFFF'

MQCCSI_* (IDs für codierte Zeichensätze)

Tabelle 54. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_DEFAULT	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_* (CICS-Header, Optionen Dialogtask)

Tabelle 55. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_* (Struktur Kanaldefinition)

Tabelle 56. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'

Tabelle 56. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_CURRENT_VERSION	9	X'00000009'
MQCD_LENGTH_4	(value differs by platform or version)	
MQCD_LENGTH_5	(value differs by platform or version)	
MQCD_LENGTH_6	(value differs by platform or version)	
MQCD_LENGTH_7	(value differs by platform or version)	
MQCD_LENGTH_8	(value differs by platform or version)	
MQCD_LENGTH_9	(value differs by platform or version)	
MQCD_CURRENT_LENGTH	(value differs by platform or version)	

MQCDC_* (Kanaldatenkonvertierung)

Tabelle 57. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION	0	X'00000000'

MQCERT_* (Typ Zertifikatprüfrichtlinie)

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_* (Funktionsmarkierungen)

Tabelle 58. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_* (CICS-Header Facility)

Tabelle 59. Konstante Namen und Werte	
Ihren Namen	**Hexadezimalwert**
MQCFAC_NONE	X'00...00' (8 Nullen)

Tabelle 59. Konstante Namen und Werte (Forts.)	
Ihren Namen	**Hexadezimalwert**
MQCFAC_NONE_ARRAY	`'\0','\0',...` (8 Nullen)

MQCFBF_* (Befehlsformat, Bytefolgefilter-Parameterstruktur)

Tabelle 60. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_* (Befehlsformat, Bytefolge-Parameterstruktur)

Tabelle 61. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFC_* (Befehlsformat, Header-Steuerungsoptionen)

Tabelle 62. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFC_LAST	1	X'00000001'
MQCFC_NOT_LAST	0	X'00000000'

MQCFGR_* (Befehlsformat, Gruppenparameterstruktur)

Tabelle 63. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_* (Befehlsformat, Headerstruktur)

Tabelle 64. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Befehlsformat, Integer-Filter Parameterstruktur)

Tabelle 65. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFIF_STRUC_LENGTH	20	X'00000014'

MQCFIL_* (Befehlsformat, Integer-Liste Parameterstruktur)

Tabelle 66. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (Befehlsformat, 64-Bit-Integer-Liste Parameterstruktur)

Tabelle 67. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_* (Befehlsformat, Integer-Parameterstruktur)

Tabelle 68. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFIN_STRUC_LENGTH	16	X'00000010'

MQCFIN64_* (Befehlsformat, 64-Bit-Integer Parameterstruktur)

Tabelle 69. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_* (Befehlsformat Repositoryaktualisierungsoptionen und Befehlsformat Warteschlangenlöschoptionen)

Befehlsformat Warteschlangenlöschoptionen

Tabelle 70. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFO_REFRESH_REPOSITORY_YES	1	X'00000001'
MQCFO_REFRESH_REPOSITORY_NO	0	X'00000000'

Befehlsformat Warteschlangenlöschoptionen

Tabelle 71. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFO_REMOVE_QUEUES_YES	1	X'00000001'
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_* (Befehlsformat Filteroperatoren)

Tabelle 72. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_GREATER	4	X'00000004'
MQCFOP_NOT_LESS	6	X'00000006'

Tabelle 72. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NOT_GREATER	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_* ('CF-Wiederherstellbarkeit')

Tabelle 73. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFR_YES	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_* (Befehlsformat, Zeichenfolgefilter-Parameterstruktur)

Tabelle 74. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_* (Befehlsformat, Zeichenfolgeliste-Parameterstruktur)

Tabelle 75. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_* (Befehlsformat, Zeichenfolge-Parameterstruktur)

Tabelle 76. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_* (Befehlsformat CF-Status)

Tabelle 77. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFSTATUS_NOT_FOUND	0	X'00000000'
MQCFSTATUS_ACTIVE	1	X'00000001'
MQCFSTATUS_IN_RECOVER	2	X'00000002'
MQCFSTATUS_IN_BACKUP	3	X'00000003'
MQCFSTATUS_FAILED	4	X'00000004'
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_UNKNOWN	6	X'00000006'

Tabelle 77. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFSTATUS_ADMIN_INCOMPLETE	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_* (Befehlsformat Strukturtypen)

Tabelle 78. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'
MQCFT_ACCOUNTING	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_* (Befehlsformat CF-Typen)

Tabelle 79. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_* (CICS-Header Funktionen)

Tabelle 80. Strukturen von Konstanten

Ihren Namen	Struktur
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	'¬','¬','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQCGWI_* (CICS-Header Abrufwarteintervall)

Tabelle 81. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_* (Automatische Kanaldefinition)

Tabelle 82. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCHAD_DISABLED	0	X'00000000'
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_* (Befehlsformat Unbestätiger Status)

Tabelle 83. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCHIDS_NOT_INDOUBT	0	X'00000000'
MQCHIDS_INDOUBT	1	X'00000001'

MQCHLD_* (Befehlsformat Kanaldispositionen)

Tabelle 84. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCHLD_ALL	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'

Tabelle 84. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHLD_SHARED	2	X'00000002'
MQCHLD_PRIVATE	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_* (Befehlsformat Kanalstatus)

Tabelle 85. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Befehlsformat, Neustartoptionen gemeinsam genutzter Kanal)

Tabelle 86. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_YES	1	X'00000001'

MQCHSR_* (Befehlsformat, Kanalstoppoptionen)

Tabelle 87. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHSR_STOP_NOT_REQUESTED	0	X'00000000'
MQCHSR_STOP_REQUESTED	1	X'00000001'

MQCHSSTATE_* (Befehlsformat Kanalteilstatus)

Tabelle 88. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHSSTATE_OTHER	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_SENDING	200	X'000000C8'
MQCHSSTATE_RECEIVING	300	X'0000012C'
MQCHSSTATE_SERIALIZING	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'

Tabelle 88. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHSSTATE_HEARTBEATING	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_CONNECTING	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_COMPRESSING	1800	X'00000708'

MQCHT_* (Kanaltypen)

Tabelle 89. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'
MQCHT_CLUSSDR	9	X'00000009'

MQCHTAB_* (Befehlsformat, Kanaltabellentypen)

Tabelle 90. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_* (Korrelations-ID)

Tabelle 91. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQCI_NONE	X'00...00' (24 Nullen)
MQCI_NONE_ARRAY	'\0','\0',... (24 Nullen)
MQCI_NEW_SESSION	X'414D5121...'

Tabelle 91. Konstante Namen und Werte (Forts.)	
Ihren Namen	**Wertschöpfung**
MQCI_NEW_SESSION_ARRAY	`'\x41','\x4D','\51','\x21',...`

MQCIH_* (CICS-Informationsheaderstruktur und Flags)

CICS-Informationsheaderstruktur

Tabelle 92. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQCIH_STRUC_ID	`"CIH¬"`
MQCIH_STRUC_ID_ARRAY	`'C','I','H','¬'`

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 93. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_CURRENT_VERSION	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

CICS-Informationsheader Flags

Tabelle 94. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCIH_NONE	0	X'00000000'
MQCIH_PASS_EXPIRATION	1	X'00000001'
MQCIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQCIH_REPLY_WITHOUT_NULLS	2	X'00000002'
MQCIH_REPLY_WITH_NULLS	0	X'00000000'
MQCIH_SYNC_ON_RETURN	4	X'00000004'
MQCIH_NO_SYNC_ON_RETURN	0	X'00000000'

MQCLCT_* (Clustercache-Typen)

Tabelle 95. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCLCT_STATIC	0	X'00000000'
MQCLCT_DYNAMIC	1	X'00000001'

MQCLRS_* (Befehlsformat Löschen Themenzeichenfolge-Bereich)

Tabelle 96. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_* (Befehlsformat Löschen Themenzeichenfolge-Typ)

Tabelle 97. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCLRT_RETAINED	1	X'00000001'

MQCLT_* (CICS-Header Verbindungstypen)

Tabelle 98. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCLT_PROGRAM	1	X'00000001'
MQCLT_TRANSACTION	2	X'00000002'

MQCLWL_* (Clusterauslastung)

Tabelle 99. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_* (Clusterübertragungs-WS-Typ)

MQCLXQ_* sind die Werte, die Sie im Warteschlangenmanagerattribut DEFCLXQ festlegen können.Das Attribut DEFCLXQ steuert, welche Übertragungswarteschlange standardmäßig von Clustersenderkanälen für den Abruf von Nachrichten ausgewählt wird, die an Clusterempfängerkanäle gesendet werden sollen.

Tabelle 100. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Zugehörige Verweise

„DefClusterXmitQueueType (MQLONG)” auf Seite 814
Das Attribut DefClusterXmitQueueType steuert, welche Übertragungswarteschlange standardmäßig von Clustersenderkanälen für den Abruf von Nachrichten ausgewählt wird, die an Clusterempfängerkanäle gesendet werden sollen.

Warteschlangenmanager ändern

Warteschlangenmanager abfragen

Inquire Queue Manager (Antwort)

„MQINQ - Objektattribute abfragen” auf Seite 700
Der Aufruf MQINQ gibt eine Ganzzahlenfeldgruppe und eine Zeichenfolgegruppe mit den Attributen eines Objekts zurück.

MQCMD_* (Befehlscodes)

Tabelle 101. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
MQCMD_CHANGE_PROCESS	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'
MQCMD_CREATE_PROCESS	5	X'00000005'
MQCMD_DELETE_PROCESS	6	X'00000006'
MQCMD_INQUIRE_PROCESS	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREATE_Q	11	X'0000000B'
MQCMD_DELETE_Q	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'
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_CHANGE_NAMELIST	32	X'00000020'
MQCMD_COPY_NAMELIST	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'

Tabelle 101. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_SUBSCRIBER	65	X'00000041'
MQCMD_REQUEST_UPDATE	66	X'00000042'
MQCMD_BROKER_INTERNAL	67	X'00000043'
MQCMD_ACTIVITY_MSG	69	X'00000045'
MQCMD_INQUIRE_CLUSTER_Q_MGR	70	X'00000046'
MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
MQCMD_SUSPEND_Q_MGR_CLUSTER	72	X'00000048'
MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_CLUSTER	74	X'0000004A'
MQCMD_TRACE_ROUTE	75	X'0000004B'
MQCMD_REFRESH_SECURITY	78	X'0000004E'
MQCMD_CHANGE_AUTH_INFO	79	X'0000004F'
MQCMD_COPY_AUTH_INFO	80	X'00000050'
MQCMD_CREATE_AUTH_INFO	81	X'00000051'
MQCMD_DELETE_AUTH_INFO	82	X'00000052'
MQCMD_INQUIRE_AUTH_INFO	83	X'00000053'
MQCMD_INQUIRE_AUTH_INFO_NAMES	84	X'00000054'
MQCMD_INQUIRE_CONNECTION	85	X'00000055'
MQCMD_STOP_CONNECTION	86	X'00000056'
MQCMD_INQUIRE_AUTH_RECS	87	X'00000057'
MQCMD_INQUIRE_ENTITY_AUTH	88	X'00000058'
MQCMD_DELETE_AUTH_REC	89	X'00000059'
MQCMD_SET_AUTH_REC	90	X'0000005A'
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'

Tabelle 101. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_COMMAND_EVENT	99	X'00000063'
MQCMD_CHANGE_SECURITY	100	X'00000064'
MQCMD_CHANGE_CF_STRUC	101	X'00000065'
MQCMD_CHANGE_STG_CLASS	102	X'00000066'
MQCMD_CHANGE_TRACE	103	X'00000067'
MQCMD_ARCHIVE_LOG	104	X'00000068'
MQCMD_BACKUP_CF_STRUC	105	X'00000069'
MQCMD_CREATE_BUFFER_POOL	106	X'0000006A'
MQCMD_CREATE_PAGE_SET	107	X'0000006B'
MQCMD_CREATE_CF_STRUC	108	X'0000006C'
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'
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_SECURITY	133	X'00000085'

Tabelle 101. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCMD_SET_ARCHIVE	134	X'00000086'
MQCMD_SET_LOG	136	X'00000088'
MQCMD_SET_SYSTEM	137	X'00000089'
MQCMD_START_CMD_SERVER	138	X'0000008A'
MQCMD_START_Q_MGR	139	X'0000008B'
MQCMD_START_TRACE	140	X'0000008C'
MQCMD_STOP_CHANNEL_INIT	141	X'0000008D'
MQCMD_STOP_CHANNEL_LISTENER	142	X'0000008E'
MQCMD_STOP_CMD_SERVER	143	X'0000008F'
MQCMD_STOP_Q_MGR	144	X'00000090'
MQCMD_STOP_TRACE	145	X'00000091'
MQCMD_SUSPEND_Q_MGR	146	X'00000092'
MQCMD_INQUIRE_CF_STRUC_NAMES	147	X'00000093'
MQCMD_INQUIRE_STG_CLASS_NAMES	148	X'00000094'
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'
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'
MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREATE_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC	173	X'000000AD'
MQCMD_INQUIRE_TOPIC	174	X'000000AE'

Tabelle 101. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Befehlsformat, Befehlsdatenwerte)

Tabelle 102. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCMDI_CMDSCOPE_ACCEPTED	1	X'00000001'
MQCMDI_CMDSCOPE_GENERATED	2	X'00000002'
MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
MQCMDI_COMMAND_ACCEPTED	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_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'
MQCMDI_SEC_TIMER_ZERO	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_* (Befehlsebenen)

Tabelle 103. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQCMDL_LEVEL_1	100
MQCMDL_LEVEL_101	101
MQCMDL_LEVEL_110	110

Tabelle 103. Konstante Namen und Werte (Forts.)	
Ihren Namen	**Wertschöpfung**
MQCMDL_LEVEL_114	114
MQCMDL_LEVEL_120	120
MQCMDL_LEVEL_200	200
MQCMDL_LEVEL_201	201
MQCMDL_LEVEL_210	210
MQCMDL_LEVEL_211	211
MQCMDL_LEVEL_220	220
MQCMDL_LEVEL_221	221
MQCMDL_LEVEL_230	230
MQCMDL_LEVEL_320	320
MQCMDL_LEVEL_420	420
MQCMDL_LEVEL_500	500
MQCMDL_LEVEL_510	510
MQCMDL_LEVEL_520	520
MQCMDL_LEVEL_530	530
MQCMDL_LEVEL_531	531
MQCMDL_LEVEL_600	600
MQCMDL_LEVEL_700	700
MQCMDL_LEVEL_701	701
MQCMDL_LEVEL_710	710
MQCMDL_LEVEL_711	711
MQCMDL_LEVEL_750	750

MQCMHO_* (Optionen und Struktur Nachrichtenkennung erstellen)

Nachrichtenkennungsoptionen und Struktur erstellen

Tabelle 104. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQCMHO_STRUC_ID	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C','M','H','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 105. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_CURRENT_VERSION	1	X'00000001'

Nachrichtenkennungsoptionen erstellen

Tabelle 106. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'
MQCMHO_VALIDATE	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_* (Verbindungsoptionen und Struktur)

Optionsstruktur für die Verbindung

Tabelle 107. Strukturen von Konstanten

Ihren Namen	Struktur
MQCNO_STRUC_ID	"CNO¬"
MQCNO_STRUC_ID_ARRAY	'C','N','O','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 108. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Verbindungsoptionen

Tabelle 109. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 109. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_ONLY	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_* (Beendigungsoptionen)

Tabelle 110. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCO_IMMEDIATE	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_* (CICS-Header, Ausgabedatenlänge)

Tabelle 111. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_* (Kanalkomprimierung)

Tabelle 112. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCOMPRESS_NOT_AVAILABLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
MQCOMPRESS_SYSTEM	8	X'00000008'

Tabelle 112. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_* (Verbindungs-ID)

Tabelle 113. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQCONNID_NONE	X'00...00' (24 Nullen)
MQCONNID_NONE_ARRAY	'\0','\0',... (24 Nullen)

MQCOPY_* (Eigenschaften-Kopierparameter)

Tabelle 114. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'
MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_* (Cluster-WS-Typen)

Tabelle 115. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTE_Q	3	X'00000003'
MQCQT_Q_MGR_ALIAS	4	X'00000004'

MQCRC_* (CICS-Header Rückkehrcodes)

Tabelle 116. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCRC_OK	0	X'00000000'
MQCRC_CICS_EXEC_ERROR	1	X'00000001'
MQCRC_MQ_API_ERROR	2	X'00000002'
MQCRC_BRIDGE_ERROR	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'
MQCRC_APPLICATION_ABEND	5	X'00000005'
MQCRC_SECURITY_ERROR	6	X'00000006'
MQCRC_PROGRAM_NOT_AVAILABLE	7	X'00000007'
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NOT_AVAILABLE	9	X'00000009'

MQCS_* (Kosumentenstatus MQCBC-Konstanten)

Tabelle 117. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDED	3	X'00000003'
MQCS_STOPPED	4	X'00000004'

MQCSC_* (CICS-Header Startcodes)

Tabelle 118. Strukturen von Konstanten

Ihren Namen	Struktur
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	'¬','¬','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQCSP_* (Verbindungssicherheitsparameter Struktur und Authentifizierungstyp)

Struktur Verbindungssicherheitsparameter

Tabelle 119. Strukturen von Konstanten

Ihren Namen	Struktur
MQCSP_STRUC_ID	"CSP¬"
MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 120. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCSP_VERSION_1	1	X'00000001'
MQCSP_CURRENT_VERSION	1	X'00000001'

Struktur Verbindungssicherheitsparameter Authentifizierungstypen

Tabelle 121. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCSP_AUTH_NONE	0	X'00000000'

Tabelle 121. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'

MQCSRV_* (Befehlsserveroptionen)

Tabelle 122. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Warteschlangenmanager Verbindungstag)

Tabelle 123. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQCT_NONE	X'00...00' (128 Nullen)
MQCT_NONE_ARRAY	'\0','\0',... (128 Nullen)

MQCTES_* (CICS-Header Taskende-Status)

Tabelle 124. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCTES_NOSYNC	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_BACKOUT	4352	X'00001100'
MQCTES_ENDTASK	65536	X'00010000'

MQCTLO_* (MQCTL-Optionen Struktur und Konsumentensteuerelement-Optionen)

Struktur MQCTL-Optionen

Tabelle 125. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 126. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQCTLO_VERSION_1	1	X'00000001'
MQCTLO_CURRENT_VERSION	1	X'00000001'

MQCTL-Optionen Konsumentensteuerelement-Optionen

Tabelle 127. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCUOWC_* (CICS-Header Arbeitseinheit-Steuerelemente)

Tabelle 128. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Struktur Kanalexitparameter)

Tabelle 129. Strukturen von Konstanten

Ihren Namen	Struktur
MQCXP_STRUC_ID	"CXP¬"
MQCXP_STRUC_ID_ARRAY	'C','X','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 130. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_CURRENT_VERSION	8	X'00000008'

MQDC_* (Zielklasse)

Tabelle 131. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQDC_MANAGED	1	X'00000001'
MQDC_PROVIDED	2	X'00000002'

MQDCC_* (Konvertierungsoptionen und Masken und Faktoren)

Konvertierungsoptionen

Tabelle 132. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQDCC_DEFAULT_CONVERSION	1	X'00000001'
MQDCC_FILL_TARGET_BUFFER	2	X'00000002'
MQDCC_INT_DEFAULT_CONVERSION	4	X'00000004'
MQDCC_SOURCE_ENC_NATIVE	(value differs by platform or version)	
MQDCC_SOURCE_ENC_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSED	32	X'00000020'
MQDCC_SOURCE_ENC_UNDEFINED	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	
MQDCC_TARGET_ENC_NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSED	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'
MQDCC_NONE	0	X'00000000'

Konvertierungsoptionen Masken und Faktoren

Tabelle 133. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Publish/Subscribe-Löschoptionen)

Tabelle 134. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL	4	X'00000004'

MQDH_* (Verteilungsheader-Struktur)

Tabelle 135. Strukturen von Konstanten

Ihren Namen	Struktur
MQDH_STRUC_ID	"DH¬¬"
MQDH_STRUC_ID_ARRAY	'D','H','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 136. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_* (Verteilungsheader-Flags)

Tabelle 137. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_* (Befehlsformat Verbindungstrennungstypen)

Tabelle 138. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_* (Verteilerlisten)

Tabelle 139. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_* (Headerstruktur nicht zustellbare Nachrichten)

Tabelle 140. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQDLH_STRUC_ID	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D','L','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 141. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDLH_VERSION_1	1	X'00000001'
MQDLH_CURRENT_VERSION	1	X'00000001'

MQDLV_* (Permanente/nicht-permanente Nachrichtenübermittlung)

Tabelle 142. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_ALL	1	X'00000001'
MQDLV_ALL_DUR	2	X'00000002'

Tabelle 142. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDLV_ALL_AVAIL	3	X'00000003'

MQDMHO_* (Optionen und Struktur Nachrichtenkennung löschen)

Optionen und Struktur Nachrichtenkennung löschen

Tabelle 143. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D','M','H','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 144. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_CURRENT_VERSION	1	X'00000001'

Optionen Nachrichtenkennung löschen

Tabelle 145. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDMHO_NONE	0	X'00000000'

MQDMPO_* (Optionen und Struktur Nachrichteneigenschaften löschen)

Optionen Struktur Nachrichteneigenschaften löschen

Tabelle 146. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 147. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_CURRENT_VERSION	1	X'00000001'

Optionen Nachrichteneigenschaften löschen

Tabelle 148. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'

Tabelle 148. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_* (DNS-Auslastungsmanagement)

Tabelle 149. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_YES	1	X'00000001'

MQDT_* (Zieltypen)

Tabelle 150. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_* (Konvertierungsexit-Parameterstruktur)

Tabelle 151. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQDXP_STRUC_ID	"DXP¬"
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 152. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_* (Signalwerte)

Tabelle 153. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQEC_MSG_ARRIVED	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED	3	X'00000003'
MQEC_WAIT_CANCELED	4	X'00000004'
MQEC_Q_MGR_QUIESCING	5	X'00000005'
MQEC_CONNECTION_QUIESCING	6	X'00000006'

MQEI_* (Ablauf)

Tabelle 154. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_* (Codierung)

MQENC_* (Codierung)

Tabelle 155. Werte von Konstanten nach Plattform

Ihren Namen	Plattform	Dezimalwert	Hexadezimalwert
MQENC_NATIVE	IBM i	273	X'00000111'
	Linux	546	X'00000222'
	Linux on SPARC	273	X'00000111'
	Linux on x86	546	X'00000222'
	Solaris on SPARC	273	X'00000111'
	UNIX-Systeme	273	X'00000111'
	Windows	546	X'00000222'
	Micro Focus COBOL unter Windows	17.	X'00000011'
	z/OS	785	X'00000311'

MQENC_* (Codierungsmasken)

Tabelle 156. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQENC_INTEGER_MASK	15	X'0000000F'
MQENC_DECIMAL_MASK	240	X'000000F0'
MQENC_FLOAT_MASK	3840	X'00000F00'
MQENC_RESERVED_MASK	-4096	X'FFFFF000'

MQENC_* (Codierungen für binäre Ganzzahlen)

Tabelle 157. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQENC_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_* (Codierungen für gepackt dezimale Ganzzahlen)

Tabelle 158. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQENC_DECIMAL_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_* (Codierungen für Fließkommazahlen)

Tabelle 159. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQENC_FLOAT_UNDEFINED	0	X'00000000'

Tabelle 159. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Headerstruktur und Flags eingebettetes Befehlsformat)

Headerstruktur eingebettetes Befehlsformat

Tabelle 160. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQEPH_STRUC_ID	"EPH¬"
MQEPH_STRUC_ID_ARRAY	'E','P','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 161. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_CURRENT_VERSION	1	X'00000001'

Header-Flags eingebettetes Befehlsformat

Tabelle 162. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_* (Befehlsformat Escape-Typen)

Tabelle 163. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQET_MQSC	1	X'00000001'

MQEVO_* (Befehlsformat Ereignisursprung)

Tabelle 164. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQEVO_OTHER	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT	2	X'00000002'
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'

MQEVR_* (Befehlsformat Ereignisaufzeichnung)

Tabelle 165. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPTION	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_* (Ablaufprüfintervall)

Tabelle 166. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQEXPI_OFF	0	X'00000000'

MQFB_* (Rückmeldungswerte)

Tabelle 167. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQFB_NONE	0	X'00000000'
MQFB_SYSTEM_FIRST	1	X'00000001'
MQFB_QUIT	256	X'00000100'
MQFB_EXPIRATION	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
MQFB_CHANNEL_COMPLETED	262	X'00000106'
MQFB_CHANNEL_FAIL_RETRY	263	X'00000107'
MQFB_CHANNEL_FAIL	264	X'00000108'
MQFB_APPL_CANNOT_BE_STARTED	265	X'00000109'
MQFB_TM_ERROR	266	X'0000010A'
MQFB_APPL_TYPE_ERROR	267	X'0000010B'
MQFB_STOPPED_BY_MSG_EXIT	268	X'0000010C'
MQFB_ACTIVITY	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
MQFB_PAN	275	X'00000113'
MQFB_NAN	276	X'00000114'
MQFB_STOPPED_BY_CHAD_EXIT	277	X'00000115'
MQFB_STOPPED_BY_PUBSUB_EXIT	279	X'00000117'
MQFB_NOT_A_REPOSITORY_MSG	280	X'00000118'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MAX_ACTIVITIES	282	X'0000011A'
MQFB_NOT_FORWARDED	283	X'0000011B'
MQFB_NOT_DELIVERED	284	X'0000011C'
MQFB_UNSUPPORTED_FORWARDING	285	X'0000011D'

Tabelle 167. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
MQFB_DATA_LENGTH_ZERO	291	X'00000123'
MQFB_DATA_LENGTH_NEGATIVE	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
MQFB_IIH_ERROR	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_INTERNAL_ERROR	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'
MQFB_CICS_CCSID_ERROR	405	X'00000195'
MQFB_CICS_ENCODING_ERROR	406	X'00000196'
MQFB_CICS_CIH_ERROR	407	X'00000197'
MQFB_CICS_UOW_ERROR	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIBER_IS_PUBLISHER	502	X'000001F6'
MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'
MQFB_SYSTEM_LAST	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_* (Befehlsformat, Erzwingungsoptionen)

Tabelle 168. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQFC_YES	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_* (Formate)

Tabelle 169. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
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','¬','¬'
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','¬','¬','¬'

Tabelle 169. Konstante Namen und Werte (Forts.)	
Ihren Namen	**Wertschöpfung**
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','¬','¬'`

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQGA_* (Gruppenattribut-Selektoren)

Tabelle 170. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQGA_FIRST	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_* (Befehlsformat, Gruppenparametertypen)

Tabelle 171. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQGACF_FIRST	8001	X'00001F41'
MQGACF_COMMAND_CONTEXT	8001	X'00001F41'
MQGACF_COMMAND_DATA	8002	X'00001F42'
MQGACF_TRACE_ROUTE	8003	X'00001F43'
MQGACF_OPERATION	8004	X'00001F44'
MQGACF_ACTIVITY	8005	X'00001F45'
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MQGACF_MESSAGE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_VALUE_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_* (Gruppen-ID)

Tabelle 172. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQGI_NONE	X'00...00' (24 Nullen)
MQGI_NONE_ARRAY	`'\0','\0',...` (24 Nullen)

MQGMO_* (Struktur und Nachrichtenabrufoptionen)

Optionsstruktur für den Nachrichtenabruf

Tabelle 173. Strukturen von Konstanten

Ihren Namen	Struktur
MQGMO_STRUC_ID	"GMO¬"
MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 174. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Nachrichtenabrufoptionen

Tabelle 175. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 175. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Gruppenstatus)

Tabelle 176. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQGS_NOT_IN_GROUP	'¬'
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQHA_* (Kennungsselektoren)

Tabelle 177. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHA_FIRST	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_* ('Bag Handles')

Tabelle 178. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_* (Verbindungskennungen)

Tabelle 179. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'

Tabelle 179. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_* (Nachrichtenkennung)

Tabelle 180. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_* (Objektkennung)

Tabelle 181. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_* (Befehlsformat Kennungsstatus)

Tabelle 182. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQHSTATE_INACTIVE	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_* (Selektoren Ganzzahlattribut)

Tabelle 183. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'

Tabelle 183. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'

Tabelle 183. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'
MQIA_INHIBIT_EVENT	48	X'00000030'
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'
MQIA_INHIBIT_SUB	182	X'000000B6'
MQIA_INTRA_GROUP_QUEUING	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	233	X'000000E9'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'

Tabelle 183. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'

Tabelle 183. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'

Tabelle 183. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_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_* (Befehlsformat, Ganzzahlparametertypen)

Tabelle 184. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQIACF_FIRST	1001	X'000003E9'
MQIACF_Q_MGR_ATTRS	1001	X'000003E9'
MQIACF_Q_ATTRS	1002	X'000003EA'
MQIACF_PROCESS_ATTRS	1003	X'000003EB'
MQIACF_NAMELIST_ATTRS	1004	X'000003EC'
MQIACF_FORCE	1005	X'000003ED'
MQIACF_REPLACE	1006	X'000003EE'
MQIACF_PURGE	1007	X'000003EF'
MQIACF_QUIESCE	1008	X'000003F0'
MQIACF_MODE	1008	X'000003F0'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_ESCAPE_TYPE	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_STATE	1028	X'00000404'
MQIACF_AUX_ERROR_DATA_INT_1	1070	X'0000042E'
MQIACF_AUX_ERROR_DATA_INT_2	1071	X'0000042F'
MQIACF_CONV_REASON_CODE	1072	X'00000430'
MQIACF_BRIDGE_TYPE	1073	X'00000431'
MQIACF_INQUIRY	1074	X'00000432'
MQIACF_WAIT_INTERVAL	1075	X'00000433'
MQIACF_OPTIONS	1076	X'00000434'
MQIACF_BROKER_OPTIONS	1077	X'00000435'
MQIACF_REFRESH_TYPE	1078	X'00000436'
MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
MQIACF_INTEGER_DATA	1080	X'00000438'
MQIACF_REGISTRATION_OPTIONS	1081	X'00000439'
MQIACF_PUBLICATION_OPTIONS	1082	X'0000043A'
MQIACF_CLUSTER_INFO	1083	X'0000043B'
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
MQIACF_Q_MGR_TYPE	1085	X'0000043D'
MQIACF_ACTION	1086	X'0000043E'
MQIACF_SUSPEND	1087	X'0000043F'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIACF_BROKER_COUNT	1088	X'00000440'
MQIACF_APPL_COUNT	1089	X'00000441'
MQIACF_ANONYMOUS_COUNT	1090	X'00000442'
MQIACF_REG_REG_OPTIONS	1091	X'00000443'
MQIACF_DELETE_OPTIONS	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_AUTHORIZATION_LIST	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_ENTITY_TYPE	1118	X'0000045E'
MQIACF_COMMAND_INFO	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SYSTEM	1122	X'00000462'
MQIACF_Q_MGR_EVENT	1123	X'00000463'
MQIACF_Q_MGR_DQM	1124	X'00000464'
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
MQIACF_QSG_DISPS	1126	X'00000466'
MQIACF_UOW_STATE	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
MQIACF_CF_STRUC_STATUS	1130	X'0000046A'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIACF_UOW_TYPE	1132	X'0000046C'
MQIACF_CF_STRUC_ATTRS	1133	X'0000046D'
MQIACF_EXCLUDE_INTERVAL	1134	X'0000046E'
MQIACF_CF_STATUS_TYPE	1135	X'0000046F'
MQIACF_CF_STATUS_SUMMARY	1136	X'00000470'
MQIACF_CF_STATUS_CONNECT	1137	X'00000471'
MQIACF_CF_STATUS_BACKUP	1138	X'00000472'
MQIACF_CF_STRUC_TYPE	1139	X'00000473'
MQIACF_CF_STRUC_SIZE_MAX	1140	X'00000474'
MQIACF_CF_STRUC_SIZE_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'
MQIACF_MOVE_TYPE	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_STATUS	1149	X'0000047D'
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_SECURITY_INTERVAL	1153	X'00000481'
MQIACF_SECURITY_SWITCH	1154	X'00000482'
MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
MQIACF_USAGE_TYPE	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
MQIACF_USAGE_PERSIST_PAGES	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
MQIACF_PAGESET_STATUS	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
MQIACF_USAGE_DATA_SET_TYPE	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_TASKS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'
MQIACF_SYSP_ARCHIVE_WTOR	1205	X'000004B5'
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_TIME_INDICATOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
MQIACF_AUTH_OPTIONS	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_ACCUMULATION	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
MQIACF_OPERATION_TYPE	1240	X'000004D8'
MQIACF_BACKOUT_COUNT	1241	X'000004D9'
MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_EXPIRY	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_MONITORING	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'
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'

Tabelle 184. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIACF_TOPIC_STATUS_TYPE	1302	X'00000516'
MQIACF_SUB_OPTIONS	1303	X'00000517'
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_CLEAR_TYPE	1305	X'00000519'
MQIACF_CLEAR_SCOPE	1306	X'0000051A'
MQIACF_SUB_LEVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE	1308	X'0000051C'
MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
MQIACF_SELECTOR_TYPE	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_LAST_USED	1351	X'00000547'

MQIACH_* (Befehlsformat, Ganzzahl-Kanaltypen)

Tabelle 185. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 185. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'
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'

Tabelle 185. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIACH_DISC_RETRY	1573	X'00000625'
MQIACH_PORT_NUMBER	1574	X'00000626'
MQIACH_HDR_COMPRESSION	1575	X'00000627'
MQIACH_MSG_COMPRESSION	1576	X'00000628'
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'
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'

Tabelle 185. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQIACH_MAX_XMIT_SIZE	1613	X'0000064D'
MQIACH_DEF_CHANNEL_DISP	1614	X'0000064E'
MQIACH_SHARING_CONVERSATIONS	1615	X'0000064F'
MQIACH_MAX_SHARING_CONVS	1616	X'00000650'
MQIACH_CURRENT_SHARING_CONVS	1617	X'00000651'
MQIACH_MAX_INSTANCES	1618	X'00000652'
MQIACH_MAX_INSTS_PER_CLIENT	1619	X'00000653'
MQIACH_CLIENT_CHANNEL_WEIGHT	1620	X'00000654'
MQIACH_CONNECTION_AFFINITY	1621	X'00000655'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'
MQIACH_MSGS_RCVD	1634	X'00000662'
MQIACH_PENDING_OUT	1635	X'00000663'
MQIACH_AVAILABLE_CIPHERSPECS	1636	X'00000664'
MQIACH_MATCH	1637	X'00000665'
MQIACH_USER_SOURCE	1638	X'00000666'
MQIACH_WARNING	1639	X'00000667'
MQIACH_DEF_RECONNECT	1640	X'00000668'
MQIACH_CHANNEL_SUMMARY_ATTRS	1642	X'0000066A'
MQIACH_LAST_USED	1642	X'0000066A'

MQIAMO_* (Befehlsformat, Integerüberwachung Parametertypen)

Tabelle 186. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQIAMO_FIRST	701	X'000002BD'
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'
MQIAMO_BACKOUTS	704	X'000002C0'
MQIAMO_BROWSES	705	X'000002C1'

Tabelle 186. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQIAMO_Q_TIME_MIN	743	X'000002E7'
MQIAMO_SETS	744	X'000002E8'

Tabelle 186. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_INTERVAL	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'
MQIAMO_FEEDBACK_MODE	793	X'00000319'
MQIAMO_RELIABILITY_TYPE	794	X'0000031A'

Tabelle 186. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_DELIVERED	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROPPED	822	X'00000336'
MQIAMO_PKTS_DUPLICATED	823	X'00000337'
MQIAMO_NACKS_CREATED	824	X'00000338'
MQIAMO_NACK_PKTS_SENT	825	X'00000339'
MQIAMO_REPAIR_PKTS_RQSTD	826	X'0000033A'
MQIAMO_REPAIR_PKTS_RCVD	827	X'0000033B'
MQIAMO_PKTS_REPAIRED	828	X'0000033C'
MQIAMO_TOTAL_MSGS_RCVD	829	X'0000033D'
MQIAMO_TOTAL_MSGS_BYTES_RCVD	830	X'0000033E'
MQIAMO_TOTAL_REPAIR_PKTS_RCVD	831	X'0000033F'
MQIAMO_TOTAL_REPAIR_PKTS_RQSTD	832	X'00000340'
MQIAMO_TOTAL_MSGS_PROCESSED	833	X'00000341'

Tabelle 186. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQIAMO_TOTAL_MSGS_SELECTED	834	X'00000342'
MQIAMO_TOTAL_MSGS_EXPIRED	835	X'00000343'
MQIAMO_TOTAL_MSGS_DELIVERED	836	X'00000344'
MQIAMO_TOTAL_MSGS_RETURNED	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

MQIAMO64_* (Befehlsformat, 64-Bit-Integerüberwachung Parametertypen)

Tabelle 187. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Integer-Systemselektoren)

Tabelle 188. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'
MQIASY_CONTROL	-5	X'FFFFFFFB'
MQIASY_COMP_CODE	-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_* (IMS-Header Authentifikator)

Tabelle 189. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQIAUT_NONE	"¬¬¬¬¬¬¬¬"
MQIAUT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQIAV_* (Ganzzahlattributwerte)

Tabelle 190. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_UNDEFINED	-2	X'FFFFFFFE'

MQICM_* (IMS-Header Festschreibungsmodus)

Tabelle 191. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_* (Befehlsformat, Optionen unbestätiger Status)

Tabelle 192. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIDO_COMMIT	1	X'00000001'
MQIDO_BACKOUT	2	X'00000002'

MQIEP_* (Schnittstelleneingangspunkte)

Struktur Verbindungssicherheitsparameter

Tabelle 193. Strukturen von Konstanten

Ihren Namen	Struktur
MQIEP_STRUC_ID	"IEP¬"
MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 194. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_* (Verwaltung Gruppenwarteschlangen)

Tabelle 195. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIGQ_DISABLED	0	X'00000000'
MQIGQ_ENABLED	1	X'00000001'

MQIGQPA_* (PUT-Berechtigung Gruppenwarteschlangen)

Tabelle 196. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIGQPA_DEFAULT	1	X'00000001'
MQIGQPA_CONTEXT	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

MQIIH_* (IMS-Informationsheaderstruktur und Flags)

IMS-Informationsheaderstruktur

Tabelle 197. Strukturen von Konstanten

Ihren Namen	Struktur
MQIIH_STRUC_ID	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 198. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

IMS-Informationsheader-Flags

Tabelle 199. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Optionen und Struktur Nachrichteneigenschaften abfragen)

Optionen Struktur Nachrichteneigenschaften abfragen

Tabelle 200. Strukturen von Konstanten

Ihren Namen	Struktur
MQIMPO_STRUC_ID	"IMPO"
MQIMPO_STRUC_ID_ARRAY	'I','M','P','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 201. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIMPO_VERSION_1	1	X'00000001'
MQIMPO_CURRENT_VERSION	1	X'00000001'

Optionen Nachrichteneigenschaften abfragen

Tabelle 202. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIMPO_CONVERT_TYPE	2	X'00000002'
MQIMPO_QUERY_LENGTH	4	X'00000004'
MQIMPO_INQ_FIRST	0	X'00000000'
MQIMPO_INQ_NEXT	8	X'00000008'
MQIMPO_INQ_PROP_UNDER_CURSOR	16	X'00000010'
MQIMPO_CONVERT_VALUE	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_* (Befehlsformat Eingangsdispositionen)

Tabelle 203. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQINBD_Q_MGR	0	X'00000000'
MQINBD_GROUP	3	X'00000003'

MQIND_* (Spezielle Indexwerte)

Tabelle 204. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_ALL	-2	X'FFFFFFFE'

MQIPADDR_* (Versionen IP-Adresse)

Tabelle 205. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_* (IMS-Header Sicherheitsbereiche)

Tabelle 206. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_* (Indextypen)

Tabelle 207. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQIT_NONE	0	X'00000000'
MQIT_MSG_ID	1	X'00000001'
MQIT_CORREL_ID	2	X'00000002'
MQIT_MSG_TOKEN	4	X'00000004'
MQIT_GROUP_ID	5	X'00000005'

MQITEM_* (Elementtyp für mqInquireItemInfo)

Tabelle 208. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQITEM_INTEGER	1	X'00000001'
MQITEM_STRING	2	X'00000002'
MQITEM_BAG	3	X'00000003'
MQITEM_BYTE_STRING	4	X'00000004'
MQITEM_INTEGER_FILTER	5	X'00000005'
MQITEM_STRING_FILTER	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
MQITEM_BYTE_STRING_FILTER	8	X'00000008'

MQITII_* (IMS-Header Transaktionsinstanz-ID)

Tabelle 209. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQITII_NONE	X'00...00' (16 Nullen)
MQITII_NONE_ARRAY	'\0','\0',... (16 Nullen)

MQITS_* (IMS-Header Transaktionsstatus)

Tabelle 210. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQITS_IN_CONVERSATION	'C'
MQITS_NOT_IN_CONVERSATION	'¬'
MQITS_ARCHITECTED	'A'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQKAI_* (KeepAlive-Intervall)

Tabelle 211. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_* (Master-Verwaltung)

Tabelle 212. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

MQMCAS_* (Befehlsformat Nachrichtenkanalagentstatus)

Tabelle 213. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMCAS_STOPPED	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_* (MCA-Typen)

Tabelle 214. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMCAT_PROCESS	1	X'00000001'
MQMCAT_THREAD	2	X'00000002'

MQMCD_* (Publish/Subscribe-Optionskennung Informationen)

Publish/Subscribe-Optionskennung Nachrichteninhaltsdeskriptor (mcd) Tags

Tabelle 215. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMCD_FOLDER_VERSION	1	X'00000001'

Publish/Subscribe-Optionskennung Befehlsnamen

Tabelle 216. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQMCD_MSG_DOMAIN	"Msd"
MQMCD_MSG_SET	"Set"
MQMCD_MSG_TYPE	"Type"
MQMCD_MSG_FORMAT	"Fmt"

Publish/Subscribe-Optionskennung XML-Befehlsnamen

Tabelle 217. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
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>"

Publish/Subscribe-Optionskennung Tagwerte

Tabelle 218. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
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_* (Nachrichtendeskriptorstruktur)

Tabelle 219. Strukturen von Konstanten

Ihren Namen	Struktur
MQMD_STRUC_ID	"MD¬¬"
MQMD_STRUC_ID_ARRAY	'M','D','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 220. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_* (Struktur Nachrichtendeskriptorerweiterung)

Tabelle 221. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQMDE_STRUC_ID	"MDE¬"
MQMDE_STRUC_ID_ARRAY	'M','D','E','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 222. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQMDE_VERSION_2	2	X'00000002'
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_* (Nachrichtendeskriptorerweiterung-Flags)

Tabelle 223. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQMDEF_NONE	0	X'00000000'

MQMDS_* (Nachrichtenübermittlungsfolge)

Tabelle 224. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQMDS_PRIORITY	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_* (Nachrichtenoptionen)

Tabelle 225. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQMF_SEGMENTATION_INHIBITED	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

MQMHBO_* (Optionen und Struktur Nachrichtenkennung an Puffer)

Struktur Optionen Nachrichtenkennung an Puffer

Tabelle 226. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQMHBO_STRUC_ID	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 227. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

Optionen Nachrichtenkennung an Puffer

Tabelle 228. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_NONE	0	X'00000000'

MQMI_* (Nachrichten-ID)

Tabelle 229. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQMI_NONE	X'00...00' (24 Nullen)
MQMI_NONE_ARRAY	'\0','\0',... (24 Nullen)

MQMMBI_* (Nachrichtenmarken-Suchintervall)

Tabelle 230. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMMBI_UNLIMITED	-1	X'FFFFFFFF'

MQMO_* (Abgleichoptionen)

Tabelle 231. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_* (Befehlsformat, Modusoptionen)

Tabelle 232. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMODE_FORCE	0	X'00000000'
MQMODE_QUIESCE	1	X'00000001'
MQMODE_TERMINATE	2	X'00000002'

MQMON_* (Überwachungswerte)

Tabelle 233. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQMON_NOT_AVAILABLE	-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_DISABLED	0	X'00000000'
MQMON_ENABLED	1	X'00000001'
MQMON_LOW	17	X'00000011'
MQMON_MEDIUM	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_* (Nachrichtentypen)

Tabelle 234. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_FIRST	65536	X'00010000'
MQMT_APPL_LAST	999999999	X'3B9AC9FF'

MQMTOK_* (Nachrichten-Token)

Tabelle 235. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQMTOK_NONE	X'00...00' (16 Nullen)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 Nullen)

MQNC_* (Namenszähler)

Tabelle 236. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_* (Klasse nicht persistente Nachrichten)

Tabelle 237. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_* (Geschwindigkeit nicht persistente Nachrichten)

Tabelle 238. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_* (Namenslistentypen)

Tabelle 239. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Namen für Name/Wert-Zeichenfolge)

Tabelle 240. Konstante Namen und Werte

Ihren Namen	Wertschöpfung
MQNVS_APPL_TYPE	"OPT_APP_GRP¬"
MQNVS_MSG_TYPE	"OPT_MSG_TYPE¬"

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQOA_* (Begrenzungen für Objektattribut-Selektoren)

Tabelle 241. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQOA_FIRST	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_* (Objektdeskriptorstruktur)

Tabelle 242. Strukturen von Konstanten

Ihren Namen	Struktur
MQOD_STRUC_ID	"OD¬¬"
MQOD_STRUC_ID_ARRAY	'O','D','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 243. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
MQOD_CURRENT_VERSION	4	X'00000004'
MQOD_CURRENT_LENGTH	(value differs by platform or version)	

MQOII_* (Objektinstanz-ID)

Tabelle 244. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQOII_NONE	X'00...00' (24 Nullen)
MQOII_NONE_ARRAY	'\0','\0',... (24 Nullen)

MQOL_* (Ursprüngliche Länge)

Tabelle 245. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOL_UNDEFINED	-1	X'FFFFFFFF'

GMQOM_* (Veraltete Db2-Nachrichtenoptionen bei Gruppenabfrage)

Tabelle 246. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOM_NO	0	X'00000000'
MQOM_YES	1	X'00000001'

MQOO_* (Öffnungsoptionen)

Tabelle 247. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Tabelle 247. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Folgende ausschließlich in C++ verwendet)

Tabelle 248. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_* (Operationscodes für MQCTL und MQCTL)

Operationscodes für MQCTL

Tabelle 249. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOP_START	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

Operationscodes für MQCB

Tabelle 250. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOP_REGISTER	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Operationscodes für MQCTL und MQCB

Tabelle 251. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOP_SUSPEND	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_* (Werte mit Bezug zur MQOPEN_PRIV-Struktur)

Tabelle 252. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_* (Aktivitätsoperationen)

Tabelle 253. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_UNKNOWN	0	X'00000000'
MQOPER_BROWSE	1	X'00000001'
MQOPER_DISCARD	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DISCARDED_PUBLISH	12	X'0000000C'
MQOPER_SYSTEM_LAST	65535	X'0000FFFF'
MQOPER_APPL_FIRST	65536	X'00010000'
MQOPER_APPL_LAST	999999999	X'3B9AC9FF'

MQOT_* (Objekttypen und erweiterte Objekttypen)

Objekttypen

Tabelle 254. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQOT_NONE	0	X'00000000'
MQOT_Q	1	X'00000001'
MQOT_NAMELIST	2	X'00000002'
MQOT_PROCESS	3	X'00000003'
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'

Tabelle 254. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQOT_CF_STRUC	10	X'0000000A'
MQOT_LISTENER	11	X'0000000B'
MQOT_SERVICE	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Erweiterte Objekttypen

Tabelle 255. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_REQUESTER_CHANNEL	1009	X'000003F1'
MQOT_RECEIVER_CHANNEL	1010	X'000003F2'
MQOT_CURRENT_CHANNEL	1011	X'000003F3'
MQOT_SAVED_CHANNEL	1012	X'000003F4'
MQOT_SVRCONN_CHANNEL	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL	1014	X'000003F6'
MQOT_SHORT_CHANNEL	1015	X'000003F7'

MQPA_* (PUT-Berechtigung)

Tabelle 256. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPA_DEFAULT	1	X'00000001'
MQPA_CONTEXT	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_* (Eigenschaftsdeskriptor, Unterstützung und Kontext)

Eigenschaftsdeskriptorstruktur

Tabelle 257. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQPD_STRUC_ID	"PD¬¬"
MQPD_STRUC_ID_ARRAY	'P','D','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 258. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Eigenschaftsdeskriptoroptionen

Tabelle 259. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPD_NONE	0	X'00000000'

Eigenschaftsunterstützungsoptionen

Tabelle 260. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Eigenschaftskontext

Tabelle 261. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPD_NO_CONTEXT	0	X'00000000'
MQPD_USER_CONTEXT	1	X'00000001'

MQPER_* (Persistente Werte)

Tabelle 262. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Plattformen)

MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
MQPL_ZOS	1	X'00000001'

MQPL_OS2	2	X'00000002'
MQPL_AIX	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
MQPL_WINDOWS	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'
MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_NSS	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
MQPL_VM	18	X'00000012'
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
MQPL_NATIVE	1	X'00000001'

MQPMO_* (Nachrichteneinreihungsoptionen und Struktur für Veröffentlichungsmaske)

Optionsstruktur für die Nachrichteneinreihung

Tabelle 263. Strukturen von Konstanten

Ihren Namen	**Struktur**
MQPMO_STRUC_ID	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 264. Werte von Konstanten

Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_CURRENT_VERSION	3	X'00000003'
MQPMO_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

Nachrichteneinreihungsoptionen

Tabelle 265. Werte von Konstanten

Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'

Tabelle 265. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'
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'

Nachrichteneinreihungsoptionen für Veröffentlichungsmaske

Tabelle 266. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_* (Nachrichteneinreihungssatz-Felder)

Tabelle 267. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPMRF_MSG_ID	1	X'00000001'
MQPMRF_CORREL_ID	2	X'00000002'
MQPMRF_GROUP_ID	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_* (Befehlsformat, Löschoptionen)

Tabelle 268. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPO_YES	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_* (Priorität)

Tabelle 269. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

MQPROP_* (Warteschlangen- und Kanal-Eigenschaftensteuerungs-Werte und maximale Eigenschaftenlänge)

Warteschlangen- und Kanal-Eigenschaftensteuerungs-Werte

Tabelle 270. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPROP_COMPATIBILITY	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Maximale Eigenschaftenlängen

Tabelle 271. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_* (PUT-Antwortwerte)

Tabelle 272. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_* (Publish/Subscribe)

Befehlsformat Publish/Subscribe-Status

Tabelle 273. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_STOPPING	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_REFUSED	6	X'00000006'

Publish/Subscribe-Tags als Zeichenfolgen

Tabelle 274. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPS_COMMAND	"MQPSCommand"	
MQPS_COMP_CODE	"MQPSCompCode"	
MQPS_CORREL_ID	"MQPSCorrelId"	
MQPS_DELETE_OPTIONS	"MQPSDelOpts"	
MQPS_ERROR_ID	"MQPSErrorId"	
MQPS_ERROR_POS	"MQPSErrorPos"	
MQPS_INTEGER_DATA	"MQPSIntData"	
MQPS_PARAMETER_ID	"MQPSParmId"	
MQPS_PUBLICATION_OPTIONS	"MQPSPubOpts"	
MQPS_PUBLISH_TIMESTAMP	"MQPSPubTime"	
MQPS_Q_MGR_NAME	"MQPSQMgrName"	
MQPS_Q_NAME	"MQPSQName"	
MQPS_REASON	"MQPSReason"	
MQPS_REASON_TEXT	"MQPSReasonText"	
MQPS_REGISTRATION_OPTIONS	"MQPSRegOpts"	
MQPS_SEQUENCE_NUMBER	"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"	
MQPS_USER_ID	"MQPSUserId"	

Publish/Subscribe-Tags als von Leerzeichen umschlossene Zeichenfolgen

Tabelle 275. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPS_COMMAND_B	"bMQPSCommandb"	
MQPS_COMP_CODE_B	"bMQPSCompCodeb"	
MQPS_CORREL_ID_B	"bMQPSCorrelIdb"	
MQPS_DELETE_OPTIONS_B	"bMQPSDelOptsb"	
MQPS_ERROR_ID_B	"bMQPSErrorIdb"	
MQPS_ERROR_POS_B	"bMQPSErrorPosb"	
MQPS_INTEGER_DATA_B	"bMQPSIntDatab"	
MQPS_PARAMETER_ID_B	"bMQPSParmIdb"	
MQPS_PUBLICATION_OPTIONS_B	"bMQPSPubOptsb"	
MQPS_PUBLISH_TIMESTAMP_B	"bMQPSPubTimeb"	
MQPS_Q_MGR_NAME_B	"bMQPSQMgrNameb"	
MQPS_Q_NAME_B	"bMQPSQNameb"	
MQPS_REASON_B	"bMQPSReasonb"	
MQPS_REASON_TEXT_B	"bMQPSReasonTextb"	
MQPS_REGISTRATION_OPTIONS_B	"bMQPSRegOptsb"	
MQPS_SEQUENCE_NUMBER_B	"bMQPSSeqNumb"	
MQPS_STREAM_NAME_B	"bMQPSStreamNameb"	
MQPS_STRING_DATA_B	"bMQPSStringDatab"	
MQPS_SUBSCRIPTION_IDENTITY_B	"bMQPSSubIdentityb"	
MQPS_SUBSCRIPTION_NAME_B	"bMQPSSubNameb"	
MQPS_SUBSCRIPTION_USER_DATA_B	"bMQPSSubUserDatab"	
MQPS_TOPIC_B	"bMQPSTopicb"	
MQPS_USER_ID_B	"bMQPSUserIdb"	

Publish/Subscribe-Befehlstagwerte als Zeichenfolgen

Tabelle 276. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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"	

Publish/Subscribe-Befehlstagwerte als von Leerzeichen umschlossene Zeichenfolgen

Tabelle 277. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPS_DELETE_PUBLICATION_B	"bDeletePubb"	
MQPS_DEREGISTER_PUBLISHER_B	"bDeregPubb"	
MQPS_DEREGISTER_SUBSCRIBER_B	"bDeregSubb"	
MQPS_PUBLISH_B	"bPublishb"	
MQPS_REGISTER_PUBLISHER_B	"bRegPubb"	
MQPS_REGISTER_SUBSCRIBER_B	"bRegSubb"	
MQPS_REQUEST_UPDATE_B	"bReqUpdateb"	

Publish/Subscribe-Optionen Tagwerte als Zeichenfolgen

Tabelle 278. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPS_ADD_NAME	"AddName"	
MQPS_ANONYMOUS	"Anon"	
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"	
MQPS_DEREGISTER_ALL	"DeregAll"	
MQPS_DIRECT_REQUESTS	"DirectReq"	
MQPS_DUPLICATES_OK	"DupsOK"	
MQPS_FULL_RESPONSE	"FullResp"	
MQPS_INCLUDE_STREAM_NAME	"InclStreamName"	
MQPS_INFORM_IF_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_ALTERATION	"NoAlter"	
MQPS_NO_REGISTRATION	"NoReg"	
MQPS_NON_PERSISTENT	"NonPers"	
MQPS_NONE	"None"	
MQPS_OTHER_SUBSCRIBERS_ONLY	"OtherSubsOnly"	
MQPS_PERSISTENT	"Pers"	
MQPS_PERSISTENT_AS_PUBLISH	"PersAsPub"	
MQPS_PERSISTENT_AS_Q	"PersAsQueue"	
MQPS_PUBLISH_ON_REQUEST_ONLY	"PubOnReqOnly"	
MQPS_RETAIN_PUBLICATION	"RetainPub"	

Tabelle 278. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPS_VARIABLE_USER_ID	"VariableUserId"	

Publish/Subscribe-Optionen Tagwerte als von Leerzeichen umschlossene Werte

Tabelle 279. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPS_ADD_NAME_B	"bAddNameb"	
MQPS_ANONYMOUS_B	"bAnonb"	
MQPS_CORREL_ID_AS_IDENTITY_B	"bCorrelAsIdb"	
MQPS_DEREGISTER_ALL_B	"bDeregAllb"	
MQPS_DIRECT_REQUESTS_B	"bDirectReqb"	
MQPS_DUPLICATES_OK_B	"bDupsOKb"	
MQPS_FULL_RESPONSE_B	"bFullRespb"	
MQPS_INCLUDE_STREAM_NAME_B	"bInclStreamNameb"	
MQPS_INFORM_IF_RETAINED_B	"bInformIfRetb"	
MQPS_IS_RETAINED_PUBLICATION_B	"bIsRetainedPubb"	
MQPS_JOIN_EXCLUSIVE_B	"bJoinExclb"	
MQPS_JOIN_SHARED_B	"bJoinSharedb"	
MQPS_LEAVE_ONLY_B	"bLeaveOnlyb"	
MQPS_LOCAL_B	"bLocalb"	
MQPS_LOCKED_B	"bLockedb"	
MQPS_NEW_PUBLICATIONS_ONLY_B	"bNewPubsOnlyb"	
MQPS_NO_ALTERATION_B	"bNoAlterb"	
MQPS_NO_REGISTRATION_B	"bNoRegb"	
MQPS_NON_PERSISTENT_B	"bNonPersb"	
MQPS_NONE_B	"bNoneb"	
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"bOtherSubsOnlyb"	
MQPS_PERSISTENT_B	"bPersb"	
MQPS_PERSISTENT_AS_PUBLISH_B	"bPersAsPubb"	
MQPS_PERSISTENT_AS_Q_B	"bPersAsQueueb"	
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"bPubOnReqOnlyb"	
MQPS_RETAIN_PUBLICATION_B	"bRetainPubb"	
MQPS_VARIABLE_USER_ID_B	"bVariableUserIdb"	

MQPSC_* (Publish/Subscribe-Optionskennung Kennungen Publish/Subscribe-Befehlsordner (psc))

Tabelle 280. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSC_FOLDER_VERSION	1	X'00000001'

MQPSC_* (Publish/Subscribe-Optionskennung Befehlsnamen)

Tabelle 281. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPSC_COMMAND	"Command"	
MQPSC_REGISTRATION_OPTION	"RegOpt"	
MQPSC_PUBLICATION_OPTION	"PubOpt"	
MQPSC_DELETE_OPTION	"DelOpt"	
MQPSC_TOPIC	"Topic"	
MQPSC_SUBSCRIPTION_POINT	"SubPoint"	
MQPSC_FILTER	"Filter"	
MQPSC_Q_MGR_NAME	"QMgrName"	
MQPSC_Q_NAME	"QName"	
MQPSC_PUBLISH_TIMESTAMP	"PubTime"	
MQPSC_SEQUENCE_NUMBER	"SeqNum"	
MQPSC_SUBSCRIPTION_NAME	"SubName"	
MQPSC_SUBSCRIPTION_IDENTITY	"SubIdentity"	
MQPSC_SUBSCRIPTION_USER_DATA	"SubUserData"	
MQPSC_CORREL_ID	"CorrelId"	

MQPSC_* (Publish/Subscribe-Optionskennung XML-Befehlsnamen)

Tabelle 282. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPSC_COMMAND_B	"<Command>"	
MQPSC_COMMAND_E	"</Command>"	
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"	
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"	
MQPSC_PUBLICATION_OPTION_B	"<PubOpt>"	
MQPSC_PUBLICATION_OPTION_E	"</PubOpt>"	
MQPSC_DELETE_OPTION_B	"<DelOpt>"	
MQPSC_DELETE_OPTION_E	"</DelOpt>"	
MQPSC_TOPIC_B	"<Topic>"	
MQPSC_TOPIC_E	"</Topic>"	
MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"	
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"	
MQPSC_FILTER_B	"<Filter>"	
MQPSC_FILTER_E	"</Filter>"	
MQPSC_Q_MGR_NAME_B	"<QMgrName>"	
MQPSC_Q_MGR_NAME_E	"</QMgrName>"	
MQPSC_Q_NAME_B	"<QName>"	
MQPSC_Q_NAME_E	"</QName>"	
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"	

Tabelle 282. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"	
MQPSC_SEQUENCE_NUMBER_B	"<SeqNum>"	
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"	
MQPSC_SUBSCRIPTION_NAME_B	"<SubName>"	
MQPSC_SUBSCRIPTION_NAME_E	"</SubName>"	
MQPSC_SUBSCRIPTION_IDENTITY_B	"<SubIdentity>"	
MQPSC_SUBSCRIPTION_IDENTITY_E	"</SubIdentity>"	
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"	
MQPSC_SUBSCRIPTION_USER_DATA_E	"</SubUserData>"	
MQPSC_CORREL_ID_B	"<CorrelId>"	
MQPSC_CORREL_ID_E	"</CorrelId>"	

MQPSC_* (Publish/Subscribe-Optionswerte als Zeichenfolgen)

Tabelle 283. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSC_DELETE_PUBLICATION	"DeletePub"	
MQPSC_DEREGISTER_SUBSCRIBER	"DeregSub"	
MQPSC_PUBLISH	"Publish"	
MQPSC_REGISTER_SUBSCRIBER	"RegSub"	
MQPSC_REQUEST_UPDATE	"ReqUpdate"	

MQPSC_* (Publish/Subscribe-Optionswerte als Zeichenfolgen)

Tabelle 284. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSC_ADD_NAME	"AddName"	
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"	
MQPSC_DEREGISTER_ALL	"DeregAll"	
MQPSC_DUPLICATES_OK	"DupsOK"	
MQPSC_FULL_RESPONSE	"FullResp"	
MQPSC_INFORM_IF_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_NON_PERSISTENT	"NonPers"	
MQPSC_OTHER_SUBS_ONLY	"OtherSubsOnly"	

Tabelle 284. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Publish/Subscribe-Optionen)

Publish/Subscribe-Optionenskennung Publish/Subscribe-Antwortordner (pscr) Tags

Tabelle 285. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSCR_FOLDER_VERSION	1	X'00000001'

Publish/Subscribe-Optionskennung Befehlsnamen

Tabelle 286. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSCR_COMPLETION	"Completion"	
MQPSCR_RESPONSE	"Response"	
MQPSCR_REASON	"Reason"	

Publish/Subscribe-Optionskennung XML-Befehlsnamen

Tabelle 287. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSCR_COMPLETION_B	"<Completion>"	
MQPSCR_COMPLETION_E	"</Completion>"	
MQPSCR_RESPONSE_B	"<Response>"	
MQPSCR_RESPONSE_E	"</Response>"	
MQPSCR_REASON_B	"<Reason>"	
MQPSCR_REASON_E	"</Reason>"	

Publish/Subscribe-Optionskennung Tagwerte

Tabelle 288. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPSCR_OK	"ok"	
MQPSCR_WARNING	"warning"	
MQPSCR_ERROR	"error"	

MQPSM_* (Publish/Subscribe-Modus)

Tabelle 289. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPSM_DISABLED	0	X'00000000'
MQPSM_COMPAT	1	X'00000001'
MQPSM_ENABLED	2	X'00000002'

MQPSPROP_* (Publish/Subscribe-Nachrichteneigenschaften)

Tabelle 290. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPAT	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

MQPSST_* (Befehlsformat Publish/Subsribe-Statustyp)

Tabelle 291. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPSST_ALL	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PARENT	2	X'00000002'
MQPSST_CHILD	3	X'00000003'

MQPUBO_* (Publish/Subscribe-Veröffentlichungsoptionen)

Tabelle 292. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLICATION	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRATION	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLICATION	16	X'00000010'

MQPXP_* (Parameterstruktur Publish/Subscribe-Routing-Exits)

Tabelle 293. Strukturen von Konstanten

Ihren Namen	Struktur
MQPXP_STRUC_ID	"PXP¬"
MQPXP_STRUC_ID_ARRAY	'P','X','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 294. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQPXP_VERSION_1	1	X'00000001'
MQPXP_CURRENT_VERSION	1	X'00000001'

MQQA_* (Warteschlangenattribute)

Get-Werte sperren

Tabelle 295. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQA_GET_INHIBITED	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

Put-Werte sperren

Tabelle 296. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQA_PUT_INHIBITED	1	X'00000001'
MQQA_PUT_ALLOWED	0	X'00000000'

Gemeinsame Nutzung der Warteschlange

Tabelle 297. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Abschottung des Systems zurücksetzen

Tabelle 298. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_BACKOUT_NOT_HARDENED	0	X'00000000'

MQQDT_* (Warteschlangendefinitionstypen)

Tabelle 299. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_* (Warteschlangen-Flags)

Tabelle 300. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

MQQMDT_* (Befehlsformat, Warteschlangenmanager-Definitionstypen)

Tabelle 301. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Warteschlangenmanager-Flags)

Tabelle 302. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Befehlsformat Warteschlangenmanager-Funktion)

Tabelle 303. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQMFAC_IMS_BRIDGE	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

MQQMSTA_* (Befehlsformat Warteschlangenmanagerstatus)

Tabelle 304. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQMSTA_STARTING	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA_QUIESCING	3	X'00000003'

MQQMT_* (Befehlsformat Warteschlangenmanagertypen)

Tabelle 305. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQMT_NORMAL	0	X'00000000'
MQQMT_REPOSITORY	1	X'00000001'

MQQO_* (Befehlsformat, Wartemodusoptionen)

Tabelle 306. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQO_YES	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQSGD_* (Merkmale für Gruppen mit gemeinsamer Warteschlange)

Tabelle 307. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
MQQSGD_GROUP	3	X'00000003'
MQQSGD_PRIVATE	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_* (Befehlsformat Status Gruppe mit gemeinsamer Warteschlange)

Tabelle 308. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQSGS_UNKNOWN	0	X'00000000'
MQQSGS_CREATED	1	X'00000001'
MQQSGS_ACTIVE	2	X'00000002'
MQQSGS_INACTIVE	3	X'00000003'
MQQSGS_FAILED	4	X'00000004'
MQQSGS_PENDING	5	X'00000005'

MQQSIE_* (Befehlsformat WS-Wartungsintervall-Ereignisse)

Tabelle 309. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_* (Befehlsformat Optionen Warteschlangenstatus 'geöffnet' für SET, BROWSE, INPUT')

Tabelle 310. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQQSO_NO	0	X'00000000'
MQQSO_YES	1	X'00000001'
MQQSO_SHARED	1	X'00000001'

Tabelle 310. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQSO_EXCLUSIVE	2	X'00000002'

MQQSOT_* (Befehlsformat, Typen Warteschlangenstatus 'geöffnet')

Tabelle 311. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQSOT_ALL	1	X'00000001'
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

MQQSUM_* (Befehlsformat, Warteschlangenstatus 'nicht festgeschriebene Nachrichten')

Tabelle 312. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQSUM_YES	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_* (Warteschlangentypen und erweiterte Warteschlangentypen)

Warteschlangentypen

Tabelle 313. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Erweiterte Warteschlangentypen

Tabelle 314. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQQT_ALL	1001	X'000003E9'

MQRC_* (Ursachencodes)

Tabelle 315. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQRC_NONE	0	X'00000000'
MQRC_APPL_FIRST	900	X'00000384'
MQRC_APPL_LAST	999	X'000003E7'
MQRC_ALIAS_BASE_Q_TYPE_ERROR	2001	X'000007D1'
MQRC_ALREADY_CONNECTED	2002	X'000007D2'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQRC_ENVIRONMENT_ERROR	2012	X'000007DC'
MQRC_EXPIRY_ERROR	2013	X'000007DD'
MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
MQRC_HCONN_ERROR	2018	X'000007E2'
MQRC_HOBJ_ERROR	2019	X'000007E3'
MQRC_INHIBIT_VALUE_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_REACHED	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_INHIBITED	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'
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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQRC_STORAGE_CLASS_ERROR	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELED	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_TRUNCATED	2120	X'00000848'
MQRC_NO_EXTERNAL_PARTICIPANTS	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_SHORTAGE	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
MQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_ERROR	2134	X'00000856'
MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_REASONS	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'
MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
MQRC_TARGET_LENGTH_ERROR	2144	X'00000860'
MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
MQRC_IIH_ERROR	2148	X'00000864'
MQRC_PCF_ERROR	2149	X'00000865'
MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
MQRC_OBJECT_RECORDS_ERROR	2155	X'0000086B'
MQRC_RESPONSE_RECORDS_ERROR	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
MQRC_PUT_MSG_RECORDS_ERROR	2159	X'0000086F'
MQRC_CONN_ID_IN_USE	2160	X'00000870'
MQRC_Q_MGR_QUIESCING	2161	X'00000871'
MQRC_Q_MGR_STOPPING	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
MQRC_TMC_ERROR	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'
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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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'
MQRC_OFFSET_ERROR	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'
MQRC_TM_ERROR	2265	X'000008D9'
MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_INHIBITED	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_CANCELED	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPPORTED	2318	X'0000090E'
MQRC_ITEM_VALUE_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_WRONG_TYPE	2326	X'00000916'
MQRC_ITEM_TYPE_ERROR	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
MQRC_CODED_CHAR_SET_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
MQRC_WIH_ERROR	2333	X'0000091D'
MQRC_RFH_ERROR	2334	X'0000091E'
MQRC_RFH_STRING_ERROR	2335	X'0000091F'
MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
MQRC_RFH_PARM_MISSING	2339	X'00000923'
MQRC_CHAR_CONVERSION_ERROR	2340	X'00000924'
MQRC_UCS2_CONVERSION_ERROR	2341	X'00000925'
MQRC_DB2_NOT_AVAILABLE	2342	X'00000926'
MQRC_OBJECT_NOT_UNIQUE	2343	X'00000927'
MQRC_CONN_TAG_NOT_RELEASED	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_REACHED	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_WRONG_CF_LEVEL	2366	X'0000093E'
MQRC_CONFIG_CREATE_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_CONFLICT	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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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'
MQRC_PROPERTY_TYPE_ERROR	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NOT_RETAINED	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CHANGED	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_CONVERTED	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_INHIBITED	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'
MQRC_XEPO_ERROR	2507	X'000009CB'
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQRC_GROUPING_NOT_ALTERABLE	2515	X'000009D3'
MQRC_SELECTOR_INVALID_FOR_TYPE	2516	X'000009D4'
MQRC_HOBJ_QUIESCED	2517	X'000009D5'
MQRC_HOBJ_QUIESCED_NO_MSGS	2518	X'000009D6'
MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
MQRC_RES_OBJECT_STRING_ERROR	2520	X'000009D8'
MQRC_CONNECTION_SUSPENDED	2521	X'000009D9'
MQRC_INVALID_DESTINATION	2522	X'000009DA'
MQRC_INVALID_SUBSCRIPTION	2523	X'000009DB'
MQRC_SELECTOR_NOT_ALTERABLE	2524	X'000009DC'
MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DELIVERED	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_STOPPED	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBITED	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_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'

Tabelle 315. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_REOPEN_EXCL_INPUT_ERROR	6100	X'000017D4'
MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
MQRC_REOPEN_TEMPORARY_Q_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED	6104	X'000017D8'
MQRC_CURSOR_NOT_VALID	6105	X'000017D9'
MQRC_ENCODING_ERROR	6106	X'000017DA'
MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_NULL_POINTER	6108	X'000017DC'
MQRC_NO_CONNECTION_REFERENCE	6109	X'000017DD'
MQRC_NO_BUFFER	6110	X'000017DE'
MQRC_BINARY_DATA_LENGTH_ERROR	6111	X'000017DF'
MQRC_BUFFER_NOT_AUTOMATIC	6112	X'000017E0'
MQRC_INSUFFICIENT_BUFFER	6113	X'000017E1'
MQRC_INSUFFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_TRUNCATED	6115	X'000017E3'
MQRC_ZERO_LENGTH	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
MQRC_NEGATIVE_OFFSET	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
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'
MQRC_WRONG_VERSION	6128	X'000017F0'
MQRC_REFERENCE_ERROR	6129	X'000017F1'

MQRCCF_* (Befehlsformat Header-Ursachencodes)

Tabelle 316. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_VALUE_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'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_NOT_REGISTERED	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_CORREL_ID_ERROR	3080	X'00000C08'
MQRCCF_NOT_AUTHORIZED	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_CONFLICT	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ACTION_VALUE_ERROR	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_NETBIOS_NAME_ERROR	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID	3096	X'00000C18'
MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'
MQRCCF_WRONG_USER	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
MQRCCF_SUB_IDENTITY_ERROR	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED	3156	X'00000C54'
MQRCCF_ALREADY_JOINED	3157	X'00000C55'
MQRCCF_OBJECT_IN_USE	3160	X'00000C58'
MQRCCF_UNKNOWN_FILE_NAME	3161	X'00000C59'
MQRCCF_FILE_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_PORT_NUMBER_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'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQRCCF_COMMAND_INHIBITED	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_OBJECT_NAME_RESTRICTED	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_EXCEEDED	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICT	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_CONFLICT	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICT	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_CONFLICT	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'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_CONFLICT	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'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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_INHIBITED	3318	X'00000CF6'
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_WRONG_CHLAUTH_TYPE	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_WRONG_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_WRONG_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_WRONG_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICT	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_EXCEEDED	3344	X'00000D10'
MQRCCF_IPADDR_ERROR	3345	X'00000D11'
MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_PROFILE_NAME_MISSING	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_WRONG_TYPE	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_WRONG_TYPE	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_WRONG_TYPE	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_REFUSED	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR	4013	X'00000FAD'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
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_PING_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_TERMINATING	4035	X'00000FC3'
MQRCCF_MQINQ_FAILED	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NOT_AVAILABLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_WRONG_CHANNEL_TYPE	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
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_WRONG_TYPE	4052	X'00000FD4'
MQRCCF_MCA_NAME_WRONG_TYPE	4053	X'00000FD5'
MQRCCF_DISC_INT_WRONG_TYPE	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_WRONG_TYPE	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_WRONG_TYPE	4056	X'00000FD8'

Tabelle 316. Werte von Konstanten (Forts.)

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRCCF_LONG_RETRY_WRONG_TYPE	4057	X'00000FD9'
MQRCCF_LONG_TIMER_WRONG_TYPE	4058	X'00000FDA'
MQRCCF_PUT_AUTH_WRONG_TYPE	4059	X'00000FDB'
MQRCCF_KEEP_ALIVE_INT_ERROR	4060	X'00000FDC'
MQRCCF_MISSING_CONN_NAME	4061	X'00000FDD'
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_NOT_AVAILABLE	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_WRONG_TYPE	4072	X'00000FE8'
MQRCCF_MR_INTERVAL_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVAL_WRONG_TYPE	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_WRONG_TYPE	4078	X'00000FEE'
MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_WRONG_TYPE	4080	X'00000FF0'
MQRCCF_CHAD_EVENT_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENT_WRONG_TYPE	4082	X'00000FF2'
MQRCCF_CHAD_EXIT_ERROR	4083	X'00000FF3'
MQRCCF_CHAD_EXIT_WRONG_TYPE	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
MQRCCF_BATCH_INT_ERROR	4086	X'00000FF6'
MQRCCF_BATCH_INT_WRONG_TYPE	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_WRONG_TYPE	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED	4095	X'00000FFF'

MQRCN_* (Konstanten für Clientverbindungswiederholung)

Tabelle 317. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_* (Empfangszeitlimit-Typen)

Tabelle 318. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_* (Vorauslesewerte)

Tabelle 319. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQREADA_NO	0	X'00000000'
MQREADA_YES	1	X'00000001'
MQREADA_DISABLED	2	X'00000002'
MQREADA_INHIBITED	3	X'00000003'
MQREADA_BACKLOG	4	X'00000004'

MQRECORDING_* (Aufzeichnungsoptionen)

Tabelle 320. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRECORDING_DISABLED	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_* (Publish/Subscribe-Registrierungsoptionen)

Tabelle 321. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 321. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_ALTERATION	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY	524288	X'00080000'
MQREGO_VARIABLE_USER_ID	1048576	X'00100000'
MQREGO_LOCKED	2097152	X'00200000'

MQRFH_* (Struktur und Flags Regel- und Formatierungsheader)

Struktur des Regel- und Formatierungsheaders

Tabelle 322. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQRFH_STRUC_ID	"RFH¬"
MQRFH_STRUC_ID_ARRAY	'R','F','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 323. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Flags Regel- und Formatierungsheader

Tabelle 324. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

MQRFH2_* (Publish/Subscribe-Optionskennung Kennungen übergeordneter RFH2-Ordner)

Tabelle 325. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (Publish/Subscribe-Optionskennung Befehlsnamen)

Tabelle 326. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRFH2_PUBSUB_CMD_FOLDER	"psc"	
MQRFH2_PUBSUB_RESP_FOLDER	"pscr"	
MQRFH2_MSG_CONTENT_FOLDER	"mcd"	
MQRFH2_USER_FOLDER	"usr"	

MQRFH2_* (Publish/Subscribe-Optionskennung XML-Befehlsnamen)

Tabelle 327. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Zurückgegebene Länge)

Tabelle 328. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRL_UNDEFINED	-1	X'FFFFFFFF'

MQRMH_* (Struktur Referenznachrichtenheader)

Tabelle 329. Strukturen von Konstanten

Ihren Namen	Struktur
MQRMH_STRUC_ID	"RMH¬"
MQRMH_STRUC_ID_ARRAY	'R','M','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 330. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRMH_VERSION_1	1	X'00000001'
MQRMH_CURRENT_VERSION	1	X'00000001'

MQRMHF_* (Flags Referenznachrichtenheader)

Tabelle 331. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRMHF_LAST	1	X'00000001'
MQRMHF_NOT_LAST	0	X'00000000'

^MQRO_* (Berichtsoptionen)

Tabelle 332. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Masken Berichtsoptionen)

Tabelle 333. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRO_REJECT_UNSUP_MASK	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_* (Traceroute)

Traceroute max. Aktivitäten (MQIACF_MAX_ACTIVITIES)

Tabelle 334. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Traceroute Details (MQIACF_ROUTE_DETAIL)

Tabelle 335. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIUM	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

Traceroute Weiterleitung (MQIACF_ROUTE_FORWARDING)

Tabelle 336. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED	512	X'00000200'
MQROUTE_FORWARD_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Traceroute Zustellung (MQIACF_ROUTE_DELIVERY)

Tabelle 337. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQROUTE_DELIVER_YES	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Traceroute Summierung (MQIACF_ROUTE_ACCUMULATION)

Tabelle 338. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQROUTE_ACCUMULATE_NONE	65539	X'00010003'
MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_AND_REPLY	65541	X'00010005'

MQRP_* (Befehlsformat, Ersetzungsoptionen)

Tabelle 339. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRP_YES	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_* (Befehlsformat Ursachenqualifikationsmerkmale)

Tabelle 340. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_STOPPING	5	X'00000005'
MQRQ_Q_MGR_QUIESCING	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_BRIDGE_STOPPED_OK	11	X'0000000B'
MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'

MQRT_* (Befehlsformat Aktualisierungstypen)

Tabelle 341. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRT_CONFIGURATION	1	X'00000001'
MQRT_EXPIRY	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_* (Nur Anforderung)

Tabelle 342. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_* (SSL-Clientauthentifizierung)

Tabelle 343. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSCA_REQUIRED	0	X'00000000'

Tabelle 343. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSCA_OPTIONAL	1	X'00000001'

MQSCO_* (SSL-Konfigurationsoptionen)

Struktur der SSL-Konfigurationsoptionen

Tabelle 344. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQSCO_STRUC_ID	"SCO¬"
MQSCO_STRUC_ID_ARRAY	'S','C','O','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 345. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
MQSCO_CURRENT_VERSION	4	X'00000004'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

SSL-Konfigurationsoptionen Schlüsselrücksetzungszähler

Tabelle 346. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Befehlsformat Warteschlangendefinitionsbereich

Tabelle 347. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_* (Veröffentlichungsbereich)

Tabelle 348. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSCOPE_ALL	0	X'00000000'
MQSCOPE_AS_PARENT	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_* (Sicherheitsfall)

Tabelle 349. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSCYC_UPPER	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_* (Objektdeskriptorstruktur)

Tabelle 350. Konstante Namen und Strukturen

Ihren Namen	Struktur
MQSD_STRUC_ID	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 351. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSD_VERSION_1	1	X'00000001'
MQSD_CURRENT_VERSION	1	X'00000001'

MQSECITEM_* (Befehlsformat Sicherheitselemente)

Tabelle 352. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSECITEM_ALL	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMDS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECSW_* (Befehlsformat Sicherheitsschalter- und Schalterstatus)

Befehlsformat Sicherheitsschalter

Tabelle 353. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSECSW_PROCESS	1	X'00000001'
MQSECSW_NAMELIST	2	X'00000002'
MQSECSW_Q	3	X'00000003'

Tabelle 353. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSECSW_TOPIC	4	X'00000004'
MQSECSW_CONTEXT	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'

Befehlsformat Sicherheitsschalter-Status

Tabelle 354. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Befehlsformat Sicherheitstypen)

Tabelle 355. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSECTYPE_AUTHSERV	1	X'00000001'
MQSECTYPE_SSL	2	X'00000002'
MQSECTYPE_CLASSES	3	X'00000003'

MQSEG_* (Segmentierung)

Tabelle 356. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQSEG_INHIBITED	'¬'
MQSEG_ALLOWED	'A'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQSEL_* (Spezialselektorwerte)

Tabelle 357. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_ANY_USER_SELECTOR	-30002	X'FFFF8ACE'

Tabelle 357. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Selektortypen)

Tabelle 358. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED	2	X'00000002'

MQSID_* (Sicherheits-ID)

Tabelle 359. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQSID_NONE	X'00...00' (40 Nullen)
MQSID_NONE_ARRAY	'\0','\0',... (40 Nullen)

MQSIDT_* (Sicherheits-ID-Typen)

Tabelle 360. Konstante Namen und Werte	
Ihren Namen	**Hexadezimalwert**
MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_* (Optionen und Struktur Nachrichteneigenschaften festlegen)

Optionen Struktur Nachrichteneigenschaften festlegen

Tabelle 361. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 362. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_CURRENT_VERSION	1	X'00000001'

Nachrichteneigenschaftsoptionen festlegen

Tabelle 363. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Subskriptions-Optionen)

Tabelle 364. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'
MQSO_GROUP_SUB	16	X'00000010'
MQSO_MANAGED	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLICATIONS_ONLY	4096	X'00001000'
MQSO_FAIL_IF_QUIESCING	8192	X'00002000'
MQSO_ALTERNATE_USER_AUTHORITY	262144	X'00040000'
MQSO_WILDCARD_CHAR	1048576	X'00100000'
MQSO_WILDCARD_TOPIC	2097152	X'00200000'
MQSO_SET_CORREL_ID	4194304	X'00400000'
MQSO_SCOPE_QMGR	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_* (Synchronisationspunktverfügbarkeit)

Tabelle 365. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

MQSQQM_* (Name Warteschlangenmanager für gemeinsame Warteschlange)

Tabelle 366. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSQQM_USE	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_* (Aktion)

Tabelle 367. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSR_ACTION_PUBLICATION	1	X'00000001'

MQSRO_* (Struktur Optionen Subskriptions-Anforderung)

Tabelle 368. Strukturen von Konstanten

Ihren Namen	Struktur
MQSRO_STRUC_ID	"SRO¬"
MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 369. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Segmentstatus)

Tabelle 370. Konstante Namen und Strukturen

Ihren Namen	Struktur
MQSS_NOT_A_SEGMENT	'¬'
MQSS_SEGMENT	'S'
MQSS_LAST_SEGMENT	'L'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQSSL_* (SSL FIPS-Voraussetzungen)

Tabelle 371. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_* (Stat-Optionen)

MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'

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

MQSTS_* (Statusberichtsstruktur)

Tabelle 372. Strukturen von Konstanten

Ihren Namen	Struktur
MQSTS_STRUC_ID	"STAT"
MQSTS_STRUC_ID_ARRAY	'S','T','A','T'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 373. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSTS_VERSION_1	1	X'00000001'
MQSTS_CURRENT_VERSION	1	X'00000001'

MQSUB_* (Permanente Subskriptionen)

Permanente Subskriptionen

Tabelle 374. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBITED	2	X'00000002'

Permanente Subskriptionen

Tabelle 375. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_* (Befehlsformat Subskriptionstypen)

Tabelle 376. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSUBTYPE_API	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
MQSUBTYPE_USER	-2	X'FFFFFFFE'

MQSUS_* (Befehlsformat Aussetzungsstatus)

Tabelle 377. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSUS_YES	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_* (Service)

Servicetypen

Tabelle 378. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSVC_TYPE_COMMAND	0	X'00000000'
MQSVC_TYPE_SERVER	1	X'00000001'

Servicesteuerungen

Tabelle 379. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL	2	X'00000002'

Servicestatus

Tabelle 380. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSVC_STATUS_STOPPED	0	X'00000000'
MQSVC_STATUS_STARTING	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
MQSVC_STATUS_STOPPING	3	X'00000003'
MQSVC_STATUS_RETRYING	4	X'00000004'

MQSYNCPOINT_* (Befehlsformat Synchronisationspunkt-Werte für Publish/Subscribe-Migration)

Tabelle 381. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSYNCPOINT_YES	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

MQSYSP_* (Befehlsformat Systemparameterwerte)

Tabelle 382. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQSYSP_NO	0	X'00000000'

Tabelle 382. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQSYSP_YES	1	X'00000001'
MQSYSP_EXTENDED	2	X'00000002'
MQSYSP_TYPE_INITIAL	10	X'0000000A'
MQSYSP_TYPE_SET	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS	13	X'0000000D'
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_BUSY	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_AVAILABLE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
MQSYSP_STATUS_COPYING_LOG	36	X'00000024'

MQTA_* (Themenattribute)

Platzhalterzeichen

Tabelle 383. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

Subskriptionen zulässig

Tabelle 384. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_INHIBITED	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

Proxy-Sub-Weitergabe

Tabelle 385. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Veröffentlichungen zulässig

Tabelle 386. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_INHIBITED	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_* (Auslösersteuerung)

Tabelle 387. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_* (TCP-Keepalive)

Tabelle 388. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_* (TCP-Stapeltypen)

Tabelle 389. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_* (Befehlsformat Zeiteinheiten)

Tabelle 390. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_* (Auslösenachrichtenstruktur)

Tabelle 391. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQTM_STRUC_ID	"TM¬¬"
MQTM_STRUC_ID_ARRAY	'T','M','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 392. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

MQTMC_* (Zeichenformatstruktur Auslösenachricht)

Tabelle 393. Strukturen von Konstanten

Ihren Namen	Struktur
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'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

MQTOPT_* (Thementyp)

Tabelle 394. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQTOPT_LOCAL	0	X'00000000'
MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_* (Automatischer Start Kanalinitiator-Trace)

Tabelle 395. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_YES	1	X'00000001'

MQTSCOPE_* (Subskriptionsumfang)

Tabelle 396. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_* (Auslösertypen)

Tabelle 397. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_* (Eigenschaften-Datentypen)

Tabelle 398. Werte von Konstanten

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

MQUA_* (Publish/Subscribe-Benutzerattributselektoren)

Tabelle 399. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQUA_FIRST	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_* (Befehlsformat Benutzer-ID-Unterstützung)

Tabelle 400. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_YES	1	X'00000001'

MQUNDELIVERED_* (Befehlsformat nicht zugestellte Werte für Publish/Subscribe-Migration)

Tabelle 401. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_* (Befehlsformat Arbeitseinheitenstatus)

Tabelle 402. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQUOWST_NONE	0	X'00000000'

Tabelle 402. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARED	2	X'00000002'
MQUOWST_UNRESOLVED	3	X'00000003'

MQUOWT_* (Befehlsformat Arbeitseinheitentypen)

Tabelle 403. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Wartenschlangenbelegung)

Tabelle 404. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION	1	X'00000001'

MQUSAGE_* (Befehlsformat Seitengruppenbelegungswerte und Datengruppenbelegungswerte)

Befehlsformat Seitengruppenbelegungswerte

Tabelle 405. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQUSAGE_PS_AVAILABLE	0	X'00000000'
MQUSAGE_PS_DEFINED	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SYSTEM	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Befehlsformat Datengruppenbelegungswerte

Tabelle 406. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Wertlänge)

Tabelle 407. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQVL_NULL_TERMINATED	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_* (Variable Benutzer-ID)

Tabelle 408. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQVU_FIXED_USER	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

MQWDR_* (Zieldatensatzstruktur Clusterauslastungsexit)

Tabelle 409. Strukturen von Konstanten

Ihren Namen	Struktur
MQWDR_STRUC_ID	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 410. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Warteintervall)

Tabelle 411. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQWI_UNLIMITED	-1	X'FFFFFFFF'

MQWIH_* (Headerstruktur und Flags Auslastungsinformationen)

Headerstruktur Auslastungsinformationen

Tabelle 412. Strukturen von Konstanten

Ihren Namen	Struktur
MQWIH_STRUC_ID	"WIH¬"
MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 413. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

Header-Flags Auslastungsinformationen

Tabelle 414. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQWIH_NONE	0	X'00000000'

MQWQR_* (WS-Datensatzstruktur Clusterauslastungsexit)

Tabelle 415. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQWQR_STRUC_ID	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 416. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Platzhalterschema)

Tabelle 417. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQWS_DEFAULT	0	X'00000000'
MQWS_CHAR	1	X'00000001'
MQWS_TOPIC	2	X'00000002'

MQWXP_* (Parameterstruktur Clusterauslastungsexit)

MQWXP_* (Parameterstruktur Clusterauslastungsexit)

Tabelle 418. Strukturen von Konstanten

Ihren Namen	Struktur
MQWXP_STRUC_ID	"WXP¬"
MQWXP_STRUC_ID_ARRAY	'W','X','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 419. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Clusterauslastungs-Flags)

Tabelle 420. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Zugehörige Verweise
Felder in MQWXP - Parameterstruktur von Exit für Clusterauslastung

MQXACT_* (API Aufrufertypen)

Tabelle 421. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_* (Exit-Befehle)

Tabelle 422. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Exit-Antworten)

Tabelle 423. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQXCC_OK	0	X'00000000'
MQXCC_SUPPRESS_FUNCTION	-1	X'FFFFFFFF'
MQXCC_SKIP_FUNCTION	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
MQXCC_FAILED	-8	X'FFFFFFF8'

MQXDR_* (Exit-Antwort)

Tabelle 424. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_* (Umgebungen)

Tabelle 425. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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_* (Struktur Optionen Registereingangspunkt und Exitoptionen)

Struktur Optionen Registereingangspunkt

Tabelle 426. Strukturen von Konstanten

Ihren Namen	Struktur
MQXEPO_STRUC_ID	"XEPO"
MQXEPO_STRUC_ID_ARRAY	'X','E','P','O'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 427. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_CURRENT_VERSION	1	X'00000001'

Exitoptionen

Tabelle 428. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQXEPO_NONE	0	X'00000000'

MQXF_* (API-Funktions-IDs)

Tabelle 429. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 429. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXF_AXUNREG	35	X'00000023'

MQXP_* (API-Cross-Exit-Parameterstruktur)

Tabelle 430. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQXP_STRUC_ID	"XP¬¬"
MQXP_STRUC_ID_ARRAY	'X','P','¬','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 431. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXP_VERSION_1	1	X'00000001'

MQXPDA_* ('Problembestimmungsbereich)

Tabelle 432. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQXPDA_NONE	X'00...00' (48 Nullen)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 Nullen)

MQXPT_* (Transporttypen)

Tabelle 433. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXPT_ALL	-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_* (Headerstruktur Übertragungswarteschlange)

Tabelle 434. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQXQH_STRUC_ID	"XQH¬"
MQXQH_STRUC_ID_ARRAY	'X','Q','H','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 435. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_* (Exit-Ursachen)

Tabelle 436. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXR_BEFORE	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_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
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_* (Exit-Antwort 2)

Tabelle 437. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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'

Tabelle 437. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_* (Exit-IDs)

Tabelle 438. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Exit-Benutzerbereichswert)

Tabelle 439. Konstante Namen und Werte	
Ihren Namen	**Wertschöpfung**
MQXUA_NONE	X'00...00' (16 Nullen)
MQXUA_NONE_ARRAY	'\0','\0',... (16 Nullen)

MQXWD_* (Exit-Wartedescriptor-Struktur)

Tabelle 440. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQXWD_STRUC_ID	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X','W','D','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 441. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQXWD_VERSION_1	1	X'00000001'

MQZAC_* (Anwendungskontextstruktur)

Tabelle 442. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQZAC_STRUC_ID	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z','A','C','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 443. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_* (Berechtigungsdatenstruktur)

Tabelle 444. Strukturen von Konstanten

Ihren Namen	Struktur
MQZAD_STRUC_ID	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 445. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_* (Entity-Typen installierbare Services)

Tabelle 446. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZAET_NONE	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
MQZAET_GROUP	2	X'00000002'
MQZAET_UNKNOWN	3	X'00000003'

MQZAO_* (Berechtigungen installierbare Services)

Tabelle 447. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
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'

Tabelle 447. Werte von Konstanten (Forts.)		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZAO_SUBSCRIBE	4096	X'00001000'
MQZAO_RESUME	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
MQZAO_CREATE	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
MQZAO_DISPLAY	262144	X'00040000'
MQZAO_ÄNDERUNG	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTHORIZE	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_ALL	16662527	X'00FE3FFF'
MQZAO_REMOVE	16777216	X'01000000'
MQZAO_NONE	0	X'00000000'

MQZAS_* (Installierbare Services - Serviceschnittstellenversion)

Tabelle 448. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
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_* (Authentifizierungstypen)

Tabelle 449. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZAT_INITIAL_CONTEXT	0	X'00000000'
MQZAT_CHANGE_CONTEXT	1	X'00000001'

MQZCI_* (Installierbare Services - Fortsetzungsanzeiger)

Tabelle 450. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_* (Entity-Datenstruktur)

Tabelle 451. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQZED_STRUC_ID	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 452. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_* (Struktur freie Parameter)

Tabelle 453. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQZFP_STRUC_ID	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 454. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZFP_VERSION_1	1	X'00000001'
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_* (Identitätskontextstruktur)

Tabelle 455. Strukturen von Konstanten	
Ihren Namen	**Struktur**
MQZIC_STRUC_ID	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

Anmerkung: Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.

Tabelle 456. Werte von Konstanten		
Ihren Namen	**Dezimalwert**	**Hexadezimalwert**
MQZIC_VERSION_1	1	X'00000001'
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_* (Funktions-IDs für Services)

Funktions-IDs aller Services

Tabelle 457. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

Funktions-IDs für Berechtigungsservice

Tabelle 458. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZID_INIT_AUTHORITY	0	X'00000000'
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY	3	X'00000003'
MQZID_DELETE_AUTHORITY	4	X'00000004'
MQZID_SET_AUTHORITY	5	X'00000005'
MQZID_GET_AUTHORITY	6	X'00000006'
MQZID_GET_EXPLICIT_AUTHORITY	7	X'00000007'
MQZID_REFRESH_CACHE	8	X'00000008'
MQZID_ENUMERATE_AUTHORITY_DATA	9	X'00000009'
MQZID_AUTHENTICATE_USER	10	X'0000000A'
MQZID_FREE_USER	11	X'0000000B'
MQZID_INQUIRE	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

Funktions-IDs für Namensservice

Tabelle 459. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZID_INIT_NAME	0	X'00000000'
MQZID_TERM_NAME	1	X'00000001'
MQZID_LOOKUP_NAME	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

Funktions-IDs für Benutzer-ID-Service

Tabelle 460. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZID_INIT_USERID	0	X'00000000'
MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_USERID	2	X'00000002'

MQZIO_* (Installierbare Services - Initialisierungsoptionen)

Tabelle 461. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECONDARY	1	X'00000001'

MQZNS_* (Namenservice - Schnittstellenversion)

Tabelle 462. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZNS_VERSION_1	1	X'00000001'

MQZSE_* (Installierbare Services - Anzeiger Aufzählungsstart)

Tabelle 463. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZSE_START	1	X'00000001'
MQZSE_CONTINUE	0	X'00000000'

MQZSL_* (Installierbare Services - Selektoranzeiger)

Tabelle 464. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZSL_NOT_RETURNED	0	X'00000000'
MQZSL_RETURNED	1	X'00000001'

MQZTO_* (Installierbare Services - Beendigungsoptionen)

Tabelle 465. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECONDARY	1	X'00000001'

MQZUS_* (Benutzer-ID-Service-Schnittstellenversion)

Tabelle 466. Werte von Konstanten

Ihren Namen	Dezimalwert	Hexadezimalwert
MQZUS_VERSION_1	1	X'00000001'

Im MQI verwendete Datentypen

Dieser Abschnitt enthält Informationen zu den Datentypen, die im Message Queue Interface (MQI) verwendet werden. Zu jedem Datentyp finden Sie Beschreibungen, Informationen zu den Feldern sowie Deklarationen für relevante Programmiersprachen.

Einführung in die Datentypen, die im Message Queue Interface verwendet werden

Dieser Abschnitt enthält eine Einführung in die in der MQI verwendeten Datentypen und gibt Ihnen Anleitungen zur Verwendung dieser Datentypen in den unterstützten Programmiersprachen.

Elementardatentypen

Dieser Abschnitt enthält Informationen zu den im Message Queue Interface (oder in den Exitfunktionen) verwendeten Datentypen. Neben einer ausführlichen Beschreibung dieser Datentypen finden Sie in den folgenden Abschnitten Beispiele zur Deklaration der Elementardatentypen in den unterstützten Programmiersprachen.

Im Message Queue Interface (oder in den Exitfunktionen) werden die folgenden Datentypen verwendet:

- Elementardatentypen oder
- Zusammenfassungen von Elementardatentypen (Feldgruppen oder Strukturen)

Im Message Queue Interface (oder in den Exitfunktionen) werden die folgenden Elementardatentypen verwendet:

Name des Elementardatentyps	Datentyp	Beschreibung
MQBOOL	Boolesch	Der Datentyp MQBOOL steht für einen booleschen Wert. Der Wert 0 bedeutet "falsch". Alle anderen Werte stehen für "true". Der Datentyp MQBOOL muss ebenso wie der Datentyp MQLONG ausgerichtet werden.

Name des Elementardatentyps	Datentyp	Beschreibung
MQBYTE	Byte	Der Datentyp MQBYTE stellt ein einzelnes Datenbyte dar. Der Bytewert wird auf keine bestimmte Weise interpretiert; er wird nicht als Binärzahl oder Zeichen, sondern als Bitfolge behandelt. Es ist keine besondere Ausrichtung erforderlich. Werden MQBYTE-Daten zwischen Warteschlangenmanagern ausgetauscht, die unterschiedliche Zeichensätze oder Codierungen verwenden, werden die MQBYTE-Daten *nicht* in irgendeiner Form konvertiert. Dasselbe gilt für die Felder *MsgId* und *CorrelId* in der MQMD-Struktur. Eine MQBYTE-Feldgruppe wird gelegentlich für die Darstellung eines Hauptspeicherbereichs verwendet, der dem Warteschlangenmanager nicht bekannt ist. Der Bereich kann beispielsweise Anwendungsnachrichtendaten oder eine Struktur enthalten. Die Ausrichtung dieses Bereichs auf Bytegrenze muss mit der Art der enthaltenen Daten kompatibel sein. In der Programmiersprache C kann für Funktionsparameter, die als Feldgruppen von MQBYTE angezeigt werden, jeder beliebige Datentyp verwendet werden. Dies ist darauf zurückzuführen, dass derartige Parameter immer nach Adressen übergeben werden und der Funktionsparameter in der Programmiersprache C als typenloser Zeiger deklariert wird.

Name des Elementardatentyps	Datentyp	Beschreibung
MQBYTEn	Zeichenfolge mit *n* Bytes	Jeder MQBYTEn-Datentyp stellt eine Zeichenfolge mit *n* Bytes dar, wobei *n* einen der folgenden Werte aufweisen kann: 8, 16, 24, 32, 40 oder 128. Jedes Byte wird durch den Datentyp MQBYTE beschrieben. Es ist keine besondere Ausrichtung erforderlich. Falls die Daten in der Bytefolge kürzer als die definierte Länge der Zeichenfolge sind, müssen die Daten bis zum Ende der Zeichenfolge mit Nullen aufgefüllt werden. Wenn der Warteschlangenmanager Bytefolgen an die Anwendung zurückgibt (beispielsweise beim Aufruf MQGET), füllt er die Zeichenfolge bis zu ihrer definierten Länge mit Nullen auf. Für die Definition der Länge von Bytefolgefeldern sind benannte Konstanten verfügbar. Diese werden im Abschnitt „Konstanten” auf Seite 51 aufgelistet.

Name des Elementardatentyps	Datentyp	Beschreibung
MQCHAR	Zeichen	Der Datentyp MQCHAR stellt ein Einzelbytezeichen oder ein Byte eines Doppelbyte- oder Mehrfachbytezeichens dar. Es ist keine besondere Ausrichtung erforderlich. Werden MQCHAR-Daten zwischen Warteschlangenmanagern ausgetauscht, die unterschiedliche Zeichensätze oder Codierungen verwenden, müssen die MQCHAR-Daten in der Regel konvertiert werden, damit die Daten ordnungsgemäß interpretiert werden können. Bei MQCHAR-Daten in der MQMD-Struktur erfolgt dies automatisch durch den Warteschlangenmanager. Die Konvertierung von MQCHAR-Daten in den Anwendungsnachrichtendaten wird durch die Option MQGMO_CONVERT gesteuert, die im Aufruf MQGET angegeben wird. Lesen Sie hierzu die Beschreibung dieser Option im Abschnitt „MQGMO – Nachrichtenabrufoptionen” auf Seite 348, die nähere Informationen enthält.

Name des Elementardatentyps	Datentyp	Beschreibung
MQCHARn	Zeichenfolge mit *n* Zeichen	Jeder MQCHARn-Datentyp stellt eine Zeichenfolge mit *n* Zeichen dar, wobei *n* einen der folgenden Werte aufweisen kann: 4, 8, 12, 20, 28, 32, 48, 64, 128 oder 256. Jedes Zeichen wird durch den Datentyp MQCHAR beschrieben. Es ist keine besondere Ausrichtung erforderlich. Wenn die Daten in der Zeichenfolge kürzer als die definierte Länge der Zeichenfolge sind, müssen sie mit Leerzeichen aufgefüllt werden, um die Zeichenfolgelänge zu erreichen. In einigen Fällen kann ein Nullzeichen verwendet werden, um die Zeichenfolge vorzeitig zu beenden, statt sie mit Leerzeichen aufzufüllen. Das Nullzeichen und die darauf folgenden Zeichen werden bis zur definierten Länge der Zeichenfolge als Leerzeichen behandelt. Die Stellen, an denen die Verwendung einer Null möglich ist, sind in den Aufruf- und Datentypbeschreibungen angegeben. Wenn der Warteschlangenmanager Zeichenfolgen an die Anwendung zurückgibt (beispielsweise beim Aufruf MQGET), füllt er die Zeichenfolge immer bis zu ihrer definierten Länge mit Leerzeichen auf. Der Warteschlangenmanager begrenzt die Zeichenfolge also nicht mit einem Nullzeichen. Für die Definition der Länge von Zeichenfolgefeldern sind benannte Konstanten verfügbar, die im Abschnitt „Konstanten” auf Seite 51 aufgelistet sind.

Name des Elementardatentyps	Datentyp	Beschreibung
MQFLOAT32	32-Bit-Gleitkommazahl	Beim Datentyp MQFLOAT32 handelt es sich um eine 32-Bit-Gleitkommazahl, deren Darstellung des Gleitkommaformats der vom Institute of Electrical and Electronics Engineers (IEEE) festgelegten Norm entspricht. Ein MQFLOAT32-Wert muss auf eine 4-Byte-Grenze ausgerichtet werden. Soll der Datentyp MQFLOAT32 in der Programmiersprache C unter z/OS verwendet werden, ist das Compiler-Flag FLOAT(IEEE) erforderlich. Die Verwendung des Datentyps MQFLOAT32 in der Programmiersprache COBOL ist auf Compiler begrenzt, die Gleitkommazahlen im IEEE-Format unterstützen. Dafür ist möglicherweise das Compiler-Flag FLOAT(NATIVE) erforderlich.
MQFLOAT64	64-Bit-Gleitkommazahl	Beim Datentyp MQFLOAT64 handelt es sich um eine 64-Bit-Gleitkommazahl, deren Darstellung des Gleitkommaformats der vom Institute of Electrical and Electronics Engineers (IEEE) festgelegten Norm entspricht. Ein MQFLOAT64-Wert muss auf eine 8-Byte-Grenze ausgerichtet werden. Soll der Datentyp MQFLOAT64 in der Programmiersprache C unter z/OS verwendet werden, ist das Compiler-Flag FLOAT(IEEE) erforderlich. Die Verwendung des Datentyps MQFLOAT64 in der Programmiersprache COBOL ist auf Compiler begrenzt, die Gleitkommazahlen im IEEE-Format unterstützen. Dafür ist möglicherweise das Compiler-Flag FLOAT(NATIVE) erforderlich.

Name des Elementardatentyps	Datentyp	Beschreibung
MQHCONFIG	Konfigurationskennung	Der Datentyp MQHCONFIG repräsentiert eine Konfigurationskennung, also die Komponente, die für einen bestimmten installierbaren Service konfiguriert wird. Eine Konfigurationskennung muss auf ihre natürliche Grenze ausgerichtet werden. Anwendungen dürfen sich nicht auf das Format der Daten verlassen, die innerhalb dieser Kennung gespeichert sind. Falls gültig, ist der zugehörige Wert für die Verwendung in weiteren MQI-Aufrufen vorgesehen, er hat neben diesem Zweck jedoch keine sonstige Bestimmung.
MQHCONN	Verbindungskennung	Der Datentyp MQHCONN stellt eine Verbindungskennung dar (also die Verbindung mit einem bestimmten Warteschlangenmanager). Eine Verbindungskennung muss auf eine 4-Byte-Grenze ausgerichtet werden. Anwendungen dürfen sich nicht auf das Format der Daten verlassen, die innerhalb dieser Kennung gespeichert sind. Falls gültig, ist der zugehörige Wert für die Verwendung in weiteren MQI-Aufrufen vorgesehen, er hat neben diesem Zweck jedoch keine sonstige Bestimmung.
MQHMSG	Nachrichtenkennung	Der Datentyp MQHMSG stellt eine Nachrichtenkennung dar, die den Zugriff auf eine Nachricht ermöglicht. Eine Nachrichtenkennung muss auf eine 8-Byte-Grenze ausgerichtet werden. Anwendungen dürfen sich nicht auf das Format der Daten verlassen, die innerhalb dieser Kennung gespeichert sind. Falls gültig, ist der zugehörige Wert für die Verwendung in weiteren MQI-Aufrufen vorgesehen, er hat neben diesem Zweck jedoch keine sonstige Bestimmung.

Name des Elementardatentyps	Datentyp	Beschreibung
MQHOBJ	Objektkennung	Der Datentyp MQHOBJ stellt eine Objektkennung dar, die den Zugriff auf ein Objekt ermöglicht. Eine Objektkennung muss auf eine 4-Byte-Grenze ausgerichtet werden. Anwendungen dürfen sich nicht auf das Format der Daten verlassen, die innerhalb dieser Kennung gespeichert sind. Falls gültig, ist der zugehörige Wert für die Verwendung in weiteren MQI-Aufrufen vorgesehen, er hat neben diesem Zweck jedoch keine sonstige Bestimmung.
MQINT8	8-Bit-Ganzzahl mit Vorzeichen	Beim Datentyp MQINT8 handelt es sich um eine 8-Bit-Ganzzahl mit Vorzeichen, die einen beliebigen Wert zwischen -128 und +127 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht.
MQINT16	16-Bit-Ganzzahl mit Vorzeichen	Beim Datentyp MQINT16 handelt es sich um eine 16-Bit-Ganzzahl mit Vorzeichen, die einen beliebigen Wert zwischen -32 768 und +32 767 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Ein MQINT16-Wert muss auf eine 2-Byte-Grenze ausgerichtet werden.
MQINT32	32-Bit-Ganzzahl mit Vorzeichen	Beim Datentyp MQINT32 handelt es sich um eine binäre 32-Bit-Ganzzahl mit Vorzeichen, die einen beliebigen Wert zwischen -2 147 483 648 und +2 147 483 647 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Weitere Informationen finden Sie in der Definition von MQLONG.

Name des Elementardatentyps	Datentyp	Beschreibung
MQINT64	64-Bit-Ganzzahl mit Vorzeichen	Beim Datentyp MQINT64 handelt es sich um eine 64-Bit-Ganzzahl mit Vorzeichen, die einen beliebigen Wert zwischen -9 223 372 036 854 775 808 und +9 223 372 036 854 775 807 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Bei COBOL ist der gültige Bereich auf -999 999 999 999 999 999 bis +999 999 999 999 999 999 beschränkt. Eine 64-Bit-Ganzzahl muss auf eine 8-Byte-Grenze ausgerichtet werden.
MQLONG	32-Bit-Ganzzahl mit Vorzeichen	Beim Datentyp MQLONG handelt es sich um eine binäre 32-Bit-Ganzzahl mit Vorzeichen, die einen beliebigen Wert zwischen -2 147 483 648 und +2 147 483 647 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Bei COBOL ist der gültige Bereich auf -999 999 999 bis +999 999 999 beschränkt. Ein MQLONG-Wert muss auf eine 4-Byte-Grenze ausgerichtet werden.
MQPID	Prozess-ID	Die WebSphere MQ-Prozess-ID. Diese Prozess-ID wird im MQ-Trace und in den FFST™-Speicherauszügen verwendet. Sie kann sich jedoch von der Betriebssystemprozess-ID unterscheiden.
MQPTR	Zeiger	Der Datentyp MQPTR ist die Adresse von Daten beliebigen Typs. Ein Zeiger muss auf seine natürliche Grenze ausgerichtet werden. Unter IBM i ist dies die 16-Byte-Grenze und auf anderen Plattformen die 8-Byte-Grenze. Typisierte Zeiger werden von einigen Programmiersprachen unterstützt; diese werden gelegentlich auch vom Message Queue Interface verwendet (beispielsweise PMQCHAR und PMQLONG in der Programmiersprache C).

Name des Elementardatentyps	Datentyp	Beschreibung
MQTID	Thread-ID	Die WebSphere MQ-Thread-ID. Diese ID wird im MQ-Trace und in den FFST™-Speicherauszügen verwendet. Sie kann sich jedoch von der Betriebssystemthread-ID unterscheiden.
MQUINT8	8-Bit-Ganzzahl ohne Vorzeichen	Beim Datentyp MQUINT8 handelt es sich um eine 8-Bit-Ganzzahl ohne Vorzeichen, die einen beliebigen Wert zwischen 0 und +255 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht.
MQUINT16	16-Bit-Ganzzahl ohne Vorzeichen	Beim Datentyp MQUINT16 handelt es sich um eine 16-Bit-Ganzzahl ohne Vorzeichen, die einen beliebigen Wert zwischen 0 und +65 535 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Ein MQINT16-Wert muss auf eine 2-Byte-Grenze ausgerichtet werden.
MQUINT32	32-Bit-Ganzzahl ohne Vorzeichen	Beim Datentyp MQUINT32 handelt es sich um eine binäre 32-Bit-Ganzzahl ohne Vorzeichen. Weitere Informationen finden Sie in der Definition von MQULONG.
MQUINT64	64-Bit-Ganzzahl ohne Vorzeichen	Beim Datentyp MQINT64 handelt es sich um eine 64-Bit-Ganzzahl ohne Vorzeichen, die einen beliebigen Wert zwischen 0 und +18 446 744 073 709 551 615 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Bei COBOL ist der gültige Bereich auf 0 bis +999 999 999 999 999 999 beschränkt. Eine 64-Bit-Ganzzahl muss auf eine 8-Byte-Grenze ausgerichtet werden.

Name des Elementardatentyps	Datentyp	Beschreibung
MQULONG	32-Bit-Ganzzahl ohne Vorzeichen	Beim Datentyp MQULONG handelt es sich um eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die einen beliebigen Wert zwischen 0 und +4 294 967 294 haben kann, sofern keine anderweitige Einschränkung durch den Kontext besteht. Bei COBOL ist der gültige Bereich auf 0 bis +999 999 999 beschränkt. Ein MQULONG-Wert muss auf eine 4-Byte-Grenze ausgerichtet werden.
PMQACH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQACH
PMQAIR	Zeiger	Zeiger auf eine Datenstruktur des Typs MQAIR
PMQAXC	Zeiger	Zeiger auf eine Datenstruktur des Typs MQAXC
PMQAXP	Zeiger	Zeiger auf eine Datenstruktur des Typs MQAXP
PMQBMHO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQBMHO
PMQBO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQBO
PMQBOOL	Zeiger	Zeiger auf Daten des Typs MQBOOL
PMQBYTE	Zeiger	Zeiger auf Daten des Typs MQBYTE
PMQBYTEn	Zeiger	Zeiger auf Daten des Typs MQBYTEn, wobei n den Wert 8, 16, 24, 32, 40 oder 128 aufweisen kann
PMQCBC	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCBC
PMQCBD	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCBD
PMQCHAR	Zeiger	Zeiger auf Daten des Typs MQCHAR
PMQCHARN	Zeiger	Zeiger auf den Datentyp MQCHARN, wobei n den Wert 4, 8, 12, 20, 28, 32, 48, 64, 128, 256 oder 264 haben kann
PMQCHARV	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCHARV
PMQCIH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCIH

Name des Elementardatentyps	Datentyp	Beschreibung
PMQCMHO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCMHO
PMQCNO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCNO
PMQCSP	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCSP
PMQCTLO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQCTLO
PMQDH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQDH
PMQDHO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQDHO
PMQDLH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQDLH
PMQDMHO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQDMHO
PMQDMPO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQDMPO
PMQEPH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQEPH
PMQFLOAT32	Zeiger	Zeiger auf eine Datenstruktur des Typs MQFLOAT32
PMQFLOAT64	Zeiger	Zeiger auf eine Datenstruktur des Typs MQFLOAT64
PMQFUNC	Zeiger	Zeiger auf eine Funktion
PMQGMO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQGMO
PMQHCONFIG	Zeiger	Zeiger auf Daten des Typs MQHCONFIG
PMQHCONN	Zeiger	Zeiger auf Daten des Typs MQHCONN
PMQHMSG	Zeiger	Zeiger auf Daten des Typs MQHMSG
PMQHOBJ	Zeiger	Zeiger auf Daten des Typs MQHOBJ
PMQIIH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQIIH
PMQIMPO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQIMPO
PMQINT8	Zeiger	Zeiger auf Daten des Typs MQINT8
PMQINT16	Zeiger	Zeiger auf Daten des Typs MQINT16

Name des Elementardatentyps	Datentyp	Beschreibung
PMQINT32	Zeiger	Zeiger auf Daten des Typs MQINT32
PMQINT64	Zeiger	Zeiger auf Daten des Typs MQINT64
PMQLONG	Zeiger	Zeiger auf Daten des Typs MQLONG
PMQMD	Zeiger	Zeiger auf eine Struktur des Typs MQMD
PMQMDE	Zeiger	Zeiger auf eine Datenstruktur des Typs MQMDE
PMQMD1	Zeiger	Zeiger auf eine Datenstruktur des Typs MQMD1
PMQMD2	Zeiger	Zeiger auf eine Datenstruktur des Typs MQMD2
PMQMHBO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQMHBO
PMQOD	Zeiger	Zeiger auf eine Datenstruktur des Typs MQOD
PMQOR	Zeiger	Zeiger auf eine Datenstruktur des Typs MQOR
PMQPD	Zeiger	Zeiger auf eine Datenstruktur des Typs MQPD
PMQPID	Zeiger	Zeiger auf eine Prozess-ID
PMQMD	Zeiger	Zeiger auf eine Datenstruktur des Typs MQMD
PMQPMO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQPMO
PMQPTR	Zeiger	Zeiger auf Daten des Typs MQPTR
PMQRFH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQRFH
PMQRFH2	Zeiger	Zeiger auf eine Datenstruktur des Typs MQRFH2
PMQRMH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQRMH
PMQRR	Zeiger	Zeiger auf eine Datenstruktur des Typs MQRR
PMQSCO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQSCO
PMQSD	Zeiger	Zeiger auf eine Datenstruktur des Typs MQSD
PMQSMPO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQSMPO

Name des Elementardatentyps	Datentyp	Beschreibung
PMQSRO	Zeiger	Zeiger auf eine Datenstruktur des Typs MQSRO
PMSSTS	Zeiger	Zeiger auf eine Datenstruktur des Typs MQSTS
PMQTID	Zeiger	Zeiger auf eine Thread-ID
PMQTM	Zeiger	Zeiger auf eine Datenstruktur des Typs MQTM
PMQTMC2	Zeiger	Zeiger auf eine Datenstruktur des Typs MQTMC2
PMQUINT8	Zeiger	Zeiger auf den Datentyp MQUINT8
PMQUINT16	Zeiger	Zeiger auf den Datentyp MQUINT16
PMQUINT32	Zeiger	Zeiger auf den Datentyp MQUINT32
PMQUINT64	Zeiger	Zeiger auf den Datentyp MQUINT64
PMQULONG	Zeiger	Zeiger auf den Datentyp MQU-LONG
PMQVOID	Zeiger	
PMQWIH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQWIH
PMQXQH	Zeiger	Zeiger auf eine Datenstruktur des Typs MQXQH

Deklarationen in der Programmiersprache C

Datentyp	Darstellung
MQBOOL	`typedef MQLONG MQBOOL;`
MQBYTE	`typedef unsigned char MQBYTE;`
MQBYTE8	`typedef MQBYTE MQBYTE8[8];`
MQBYTE16	`typedef MQBYTE MQBYTE16[16];`
MQBYTE24	`typedef MQBYTE MQBYTE24[24];`
MQBYTE32	`typedef MQBYTE MQBYTE32[32];`
MQBYTE40	`typedef MQBYTE MQBYTE40[40];`
MQCHAR	`typedef char MQCHAR;`

Datentyp	Darstellung
MQCHAR4	`typedef MQCHAR MQCHAR4[4];`
MQCHAR8	`typedef MQCHAR MQCHAR8[8];`
MQCHAR12	`typedef MQCHAR MQCHAR12[12];`
MQCHAR20	`typedef MQCHAR MQCHAR20[20];`
MQCHAR28	`typedef MQCHAR MQCHAR28[28];`
MQCHAR32	`typedef MQCHAR MQCHAR32[32];`
MQCHAR48	`typedef MQCHAR MQCHAR48[48];`
MQCHAR64	`typedef MQCHAR MQCHAR64[64];`
MQCHAR128	`typedef MQCHAR MQCHAR128[128];`
MQCHAR256	`typedef MQCHAR MQCHAR256[256];`
MQFLOAT32	`typedef float MQFLOAT32;`
MQFLOAT64	`typedef double MQFLOAT64;`
MQHCONFIG	`typedef void MQPOINTER MQHCONFIG;`
MQHCONN	`typedef MQLONG MQHCONN;`
MQHOBJ	`typedef MQLONG MQHOBJ;`
MQINT8	`typedef signed char MQINT8;`
MQINT16	`typedef short MQINT16;`

Datentyp	Darstellung

MQINT64	Auf 64-Bit-UNIX-Systemen:
	`typedef long;`
	Auf 32-Bit-AIX-, Solaris- und HP-UX-Systemen:
	`typedef int64_t;`
	Unter IBM i, Linuxund z/OS:
	`typedef long long;`
	Unter Windows:
	`typedef _int64;`

MQLONG	Unter IBM i:
	`typedef long MQLONG;`
	Sonstige Plattformen:
	`if defined(MQ_64_BIT)`
	`    typedef int MQLONG;`
	`else`
	`    typedef long MQLONG;`

MQPID	`typedef MQLONG MQPID;`
MQPTR	`typedef void MQPOINTER MQPTR;`
MQTID	`typedef MQLONG MQTID;`
MQUINT8	`typedef unsigned char MQUINT8;`
MQUINT16	`typedef unsigned short MQUINT16;`

MQUINT64	Auf 64-Bit-UNIX-Systemen:
	`typedef unsigned long;`
	Auf 32-Bit-AIX-, Solaris- und HP-UX-Systemen:
	`typedef uint64_t;`
	Unter IBM i, Linuxund z/OS:
	`typedef unsigned long long;`
	Unter Windows:
	`typedef unsigned _int64;`

Datentyp	Darstellung

MQULONG	Unter IBM i:
	`typedef unsigned long MQULONG;`
	Sonstige Plattformen:
	`if defined(MQ_64_BIT)`
	`  typedef unsigned int MQULONG;`
	`else`
	`  typedef unsigned long MQULONG;`

PMQBO	`typedef MQBO MQPOINTER PMQBO;`
PMQBOOL	`typedef MQBOOL MQPOINTER PMQBOOL;`
PMQBYTE	`typedef MQBYTE MQPOINTER PMQBYTE;`
PMQBYTE8	`typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];`
PMQBYTE16	`typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];`
PMQBYTE24	`typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];`
PMQBYTE32	`typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];`
PMQBYTE40	`typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];`
PMQBYTE128	`typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];`
PMQCHAR	`typedef MQCHAR MQPOINTER PMQCHAR;`
PMQCHAR4	`typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];`
PMQCHAR8	`typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];`
PMQCHAR12	`typedef MQCHAR12[12] MQPOINTER PMQCHAR12[12];`
PMQCHAR20	`typedef MQCHAR20[20] MQPOINTER PMQCHAR20[20];`
PMQCHAR28	`typedef MQCHAR28[28] MQPOINTER PMQCHAR28[28];`
PMQCHAR32	`typedef MQCHAR32[32] MQPOINTER PMQCHAR32[32];`
PMQCHAR48	`typedef MQCHAR48[48] MQPOINTER PMQCHAR48[48];`
PMQCHAR64	`typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];`

Datentyp	Darstellung
PMQCHAR128	`typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];`
PMQCHAR256	`typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];`
PMQCHAR264	`typedef MQCHAR264[264] MQPOINTER PMQCHAR264[264];`
PMQCIH	`typedef MQCIH MQPOINTER PMQCIH;`
PMQCNO	`typedef MQCNO MQPOINTER PMQCNO;`
PMQDLH	`typedef MQDLH MQPOINTER PMQDLH;`
PMQFUNC	`typedef void MQPOINTER PMQFUNC;`
PMQFLOAT32	`typedef MQFLOAT32 MQPOINTER PMQFLOAT32;`
PMQFLOAT64	`typedef MQFLOAT64 MQPOINTER PMQFLOAT64;`
PMQGMO	`typedef MQGMO MQPOINTER PMQGMO;`
PMQHCONFIG	`typedef MQHCONFIG MQPOINTER PMQHCONFIG;`
PMQHCONN	`typedef MQHCONN MQPOINTER PMQHCONN;`
PMQHOBJ	`typedef MQHOBJ MQPOINTER PMQHOBJ;`
PMQIIH	`typedef MQIIH MQPOINTER PMQIIH;`
PMQINT8	`typedef MQINT8 MQPOINTER PMQINT8;`
PMQINT16	`typedef MQINT16 MQPOINTER PMQINT16;`
PMQLONG	`typedef MQLONG MQPOINTER PMQLONG;`
PMQMD	`typedef MQMD MQPOINTER PMQMD;`
PMQMD1	`typedef MQMD1[1] MQPOINTER PMQMD1[1];`
PMQMDE	`typedef MQMDE MQPOINTER PMQMDE;`
PMQOD	`typedef MQOD MQPOINTER PMQOD;`

Datentyp	Darstellung
PMQPMO	`typedef MQPMO MQPOINTER PMQPMO;`
PMQPTR	`typedef MQPTR MQPOINTER PMQPTR;`
PMQRFH	`typedef MQRFH MQPOINTER PMQRFH;`
PMQRFH2	`typedef MQRFH2[2] MQPOINTER PMQRFH2[2];`
PMQRMH	`typedef MQRMH MQPOINTER PMQRMH;`
PMQTM	`typedef MQTM MQPOINTER PMQTM;`
PMQTMC2	`typedef MQTMC2[2] MQPOINTER PMQTMC2[2];`
PMQUINT8	`typedef MQUINT8 MQPOINTER PMQUINT8;`
PMQUINT16	`typedef MQUINT16 MQPOINTER PMQUINT16;`
PMQULONG	`typedef MQULONG MQPOINTER PMQULONG;`
PMQVOID	`typedef void MQPOINTER PMQVOID;`
PMQWIH	`typedef MQWIH MQPOINTER PMQWIH;`
PMQXQH	`typedef MQXQH MQPOINTER PMQXQH;`
PPMQBO	`typedef PMQBO MQPOINTER PPMQBO;`
PPMQBYTE	`typedef PMQBYTE MQPOINTER PPMQBYTE;`
PPMQCHAR	`typedef PMQCHAR MQPOINTER PPMQCHAR;`
PPMQCNO	`typedef PMQCNO MQPOINTER PPMQCNO;`
PPMQGMO	`typedef PMQGMO MQPOINTER PPMQGMO;`
PPMQHCONN	`typedef PMQHCONN MQPOINTER PPMQHCONN;`
PPMQHOBJ	`typedef PMQHOBJ MQPOINTER PPMQHOBJ;`
PPMQLONG	`typedef PMQLONG MQPOINTER PPMQLONG;`

Datentyp	Darstellung
PPMQMD	`typedef PMQMD MQPOINTER PPMQMD;`
PPMQOD	`typedef PMQOD MQPOINTER PPMQOD;`
PPMQPMO	`typedef PMQPMO MQPOINTER PPMQPMO;`
PPMQULONG	`typedef PMQULONG MQPOINTER PPMQULONG;`
PPMQVOID	`typedef PMQVOID MQPOINTER PPMQVOID;`
Dabei steht `defined(MQ_64_BIT)` für eine 64-Bit-Plattform.	

Eine Beschreibung der Makrovariablen MQPOINTER finden Sie im Abschnitt „Datentypen” auf Seite 246.

Deklarationen in der Programmiersprache COBOL

Datentyp	Darstellung
MQBOOL	`PIC S9(9) BINARY`
MQBYTE	`PIC X`
MQBYTE8	`PIC X(8)`
MQBYTE16	`PIC X(16)`
MQBYTE24	`PIC X(24)`
MQBYTE32	`PIC X(32)`
MQBYTE40	`PIC X(40)`
MQCHAR	`PIC X`
MQCHAR4	`PIC X(4)`
MQCHAR8	`PIC X(8)`
MQCHAR12	`PIC X(12)`
MQCHAR20	`PIC X(20)`
MQCHAR28	`PIC X(28)`

Datentyp	Darstellung
MQCHAR32	`PIC X(32)`
MQCHAR48	`PIC X(48)`
MQCHAR64	`PIC X(64)`
MQCHAR128	`PIC X(128)`
MQCHAR256	`PIC X(256)`
MQFLOAT32	`USAGE COMP-1`
MQFLOAT64	`USAGE COMP-2`
MQHCONN	`PIC S9(9) BINARY`
MQHOBJ	`PIC S9(9) BINARY`
MQINT8	`PIC S9(2) BINARY`
MQINT16	`PIC S9(4) BINARY`
MQINT64	`PIC S9(18) BINARY`
MQLONG	`PIC S9(9) BINARY`
MQPTR	`POINTER`
MQUINT8	`PIC 9(2) BINARY`
MQUINT16	`PIC 9(4) BINARY`
MQUINT64	`PIC 9(18) BINARY`
MQULONG	`PIC 9(9) BINARY`

Deklarationen in der Programmiersprache PL/I
PL/I wird unter z/OS unterstützt.

Datentyp	Darstellung
MQBOOL	`fixed bin(31)`

Datentyp	Darstellung
MQBYTE	`char(1)`
MQBYTE8	`char(8)`
MQBYTE16	`char(16)`
MQBYTE24	`char(24)`
MQBYTE32	`char(32)`
MQBYTE40	`char(40)`
MQCHAR	`char(1)`
MQCHAR4	`char(4)`
MQCHAR8	`char(8)`
MQCHAR12	`char(12)`
MQCHAR20	`char(20)`
MQCHAR28	`char(28)`
MQCHAR32	`char(32)`
MQCHAR48	`char(48)`
MQCHAR64	`char(64)`
MQCHAR128	`char(128)`
MQCHAR256	`char(256)`
MQFLOAT32	`binary float(21) ieee`
MQFLOAT64	`binary float(52) ieee`
MQHCONN	`fixed bin(31)`
MQHOBJ	`fixed bin(31)`

Datentyp	Darstellung
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)`

Deklarationen in der System/390-Assemblersprache
Die Assemblersprache für System/390 wird nur unter z/OS unterstützt.

Datentyp	Darstellung
MQBOOL	`DS F`
MQBYTE	`DS XL1`
MQBYTE8	`DS XL8`
MQBYTE16	`DS XL16`
MQBYTE24	`DS XL24`
MQBYTE32	`DS XL32`
MQBYTE40	`DS XL40`
MQCHAR	`DS CL1`
MQCHAR4	`DS CL4`
MQCHAR8	`DS CL8`

Datentyp	Darstellung
MQCHAR12	`DS CL12`
MQCHAR20	`DS CL20`
MQCHAR28	`DS CL28`
MQCHAR32	`DS CL32`
MQCHAR48	`DS CL48`
MQCHAR64	`DS CL64`
MQCHAR128	`DS CL128`
MQCHAR256	`DS CL256`
MQFLOAT32	`DS EB`
MQFLOAT64	`DS DB`
MQHCONN	`DS F`
MQHOBJ	`DS F`
MQINT8	`DS XL1`
MQINT16	`DS H`
MQINT64	`DS D`
MQLONG	`DS F`
MQPTR	`DS F`
MQUINT8	`DS XL1`
MQUINT16	`DS H`
MQUINT64	`DS D`
MQULONG	`DS F`

Strukturdatentypen - Einführung

Dieser Abschnitt enthält eine Einführung in die Strukturdatentypen, die im Message Queue Interface (MQI) verwendet werden. Die Strukturdatentypen selbst werden in nachfolgenden Abschnitten beschrieben.

Zusammenfassung

Die folgenden Tabellen enthalten eine Übersicht über die im Message Queue Interface verwendeten Strukturdatentypen.

Tabelle 467. In MQI-Aufrufen (oder in Exitfunktionen) verwendete Strukturdatentypen

Struktur	**Beschreibung**	**Mögliche Aufrufe**
MQACH	Header für API-Exitkette	
MQAIR	Datensatz mit Authentifizierungsdaten	MQCONNX
MQAXC	API-Exitkontext	
MQAXP	API-Exitparameter	
MQBMHO	Puffer Nachrichtenkennung, Optionen	MQBUFMH
MQBO	Startoptionen	MQBEGIN
MQCBD	Callback-Deskriptor	MQCB
MQCBO	Optionen für Behältererstellung	mqCreateBag
MQCHARV	Zeichenfolge variabler Länge	MQINQMP
MQCNO	Verbindungsoptionen	MQCONNX
MQCSP	Sicherheitsparameter	MQCONNX
MQCTLO	Callback-Optionen	MQCTL
MQDMPO	Optionen für das Löschen von Nachrichteneigenschaften	MQDLTMP
MQGMO	Optionen für den Nachrichtenabruf	MQGET
MQIMPO	Optionen für das Abfragen von Nachrichteneigenschaften	MQINQMP
MQMD	Nachrichtendeskriptor	MQBUFMH, MQMHBUF, MQCB,MQGET, MQPUT, MQPUT1
MQMHBO	Nachrichtenkennung für Puffer, Optionen	MQMHBUF
MQOD	Objektdeskriptor	MQOPEN, MQPUT1
MQOR	Objektdatensatz	MQOPEN, MQPUT1
MQPD	Eigenschaftsdeskriptor	MQSETMP
MQPMO	Optionen für die Nachrichteneinreihung	MQPUT, MQPUT1
MQPMR	Datensatz für die Nachrichteneinreihung	MQPUT, MQPUT1

Tabelle 467. In MQI-Aufrufen (oder in Exitfunktionen) verwendete Strukturdatentypen (Forts.)		
Struktur	**Beschreibung**	**Mögliche Aufrufe**
MQRR	Antwortdatensatz	MQOPEN, MQPUT, MQPUT1
MQSCO	SSL-Konfigurationsoptionen	MQCONNX
MQSD	Subskriptionsdeskriptor	MQSUB
MQSMPO	Option für die Festlegung von Nachrichteneigenschaften	MQSETMP
MQSRO	Optionen für Subskriptionsanforderung	MQSUBRQ
MQSTS	Struktur von Statusberichten	MQSTAT

Tabelle 468. In Nachrichtendaten verwendete Strukturdatentypen:	
Struktur	**Beschreibung**
MQCIH	CICS-Informationsheader
MQCFH	PCF-Header
MQEPH	Eingebetteter PCF-Header
MQDH	Verteilungsheader
MQDLH	Header für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten)
MQIIH	IMS-Informationsheader
MQMDE	Nachrichtendeskriptorerweiterung
MQRFH	Header für Regeln und Formatierung
MQRFH2	Header 2 für Regeln und Formatierung
MQRMH	Header für Referenznachrichten
MQTM	Auslösenachricht
MQTMC2	Auslösenachricht (Zeichenformat 2)
MQWIH	Auslastungs-Header
MQXQH	Header der Übertragungswarteschlange

Anmerkung: Die Struktur MQDXP (Parameter des Datenkonvertierungsexits) wird gemeinsam mit den zugehörigen Datenkonvertierungsaufrufen im Abschnitt „Datenkonvertierungsexit” auf Seite 908 beschrieben.

Regeln für Strukturdatentypen

Da die Programmiersprachen hinsichtlich der jeweiligen Unterstützungsstufe für Strukturen variieren, wurden gewisse Regeln und Konventionen eingeführt, damit die MQI-Strukturen in den einzelnen Programmiersprachen auf einheitliche Weise zugeordnet werden können:

1. Strukturen müssen auf ihre natürliche Grenze ausgerichtet werden.
   - Für die meisten MQI-Strukturen ist eine 4-Byte-Ausrichtung erforderlich.
   - Unter IBM i ist für Strukturen, die Zeiger enthalten, eine 16-Byte-Ausrichtung erforderlich. Dabei handelt es sich um die Strukturen MQCNO, MQOD und MQPMO.
2. Jedes Feld in einer Struktur muss auf seine natürliche Grenze ausgerichtet werden.

- Felder mit Datentypen, die mit MQLONG gleichgesetzt werden, müssen auf 4-Byte-Grenzen ausgerichtet werden.
- Felder mit Datentypen, die mit MQPTR gleichgesetzt werden, müssen unter IBM i auf 16-Byte-Grenzen und in anderen Umgebungen auf 4-Byte-Grenzen ausgerichtet werden.
- Sonstige Felder werden auf 1-Byte-Grenzen ausgerichtet.

3. Die Länge einer Struktur muss ein Vielfaches ihrer Ausrichtung auf Bytegrenze betragen.
   - Die meisten MQI-Strukturen haben Längen, die ein Vielfaches von 4 Bytes sind.
   - Unter IBM i haben Strukturen mit Zeigern Längen, die ein Vielfaches von 16 Byte sind.
4. Sofern erforderlich, müssen Füllbytes oder -felder hinzugefügt werden, damit die obigen Regeln eingehalten werden.

In den Beschreibungen verwendete Konventionen

Die Beschreibung jedes Strukturdatentyps umfasst Folgendes:

- Übersicht über den Zweck und die Verwendung der Struktur
- Beschreibungen der Felder in der Struktur, die unabhängig von der jeweiligen Programmiersprache gelten
- Beispiele für die Deklaration der Struktur in den einzelnen unterstützten Programmiersprachen

Die Beschreibung jedes einzelnen Strukturdatentyps enthält folgende Abschnitte:

Name der Struktur
Der Name der Struktur, dem eine Zusammenfassung der Felder in der Struktur folgt.

Übersicht
Eine Kurzbeschreibung des Zwecks und der Verwendung der Struktur.

Felder
Beschreibungen der Felder. Für jedes Feld folgt auf den Namen des Felds der elementare Datentyp in runden Klammern (). Im Text werden Feldnamen in Kursivschrift angezeigt, z. B. *Version*.

Außerdem wird der Zweck des Felds in einer Beschreibung erläutert, und es wird eine Liste aller Werte bereitgestellt, die für das Feld möglich sind. Namen von Konstanten werden in Großbuchstaben angegeben. Beispiel: MQGMO_STRUC_ID. Mehrere Konstanten mit dem gleichen Präfix werden mit einem Stern (*) angegeben. Beispiel: MQIA_*.

In den Feldbeschreibungen werden die folgenden Begriffe verwendet:

Eingabe
Sie übergeben Informationen im Feld, wenn Sie den Aufruf ausführen.

Ausgabe
Der Warteschlangenmanager gibt Informationen im Feld zurück, wenn der Aufruf beendet oder fehlgeschlagen ist.

Eingabe/Ausgabe
Sie übergeben Informationen im Feld, wenn Sie einen Aufruf ausführen, und der Warteschlangenmanager ändert die Informationen, wenn der Aufruf beendet oder fehlgeschlagen ist.

Anfangswert
Eine Tabelle mit den Anfangswerten der einzelnen Felder in den Datendefinitionsdateien, die vom Message Queue Interface bereitgestellt werden.

Deklaration in Programmiersprache C
Typische Deklaration der Struktur in der Programmiersprache C.

COBOL-Delaration
Typische Deklaration der Struktur in der Programmiersprache COBOL.

Deklaration in PL/I
Typische Deklaration der Struktur in der Programmiersprache PL/I.

Deklaration in System/390 Assembler
Typische Deklaration der Struktur in der System/390-Assemblersprache.

Deklaration in Visual Basic
Typische Deklaration der Struktur in Visual Basic.

C-Programmierung

Dieser Abschnitt enthält Informationen, die Sie bei der Verwendung der MQI aus der Programmiersprache C unterstützen.

Headerdateien
Headerdateien werden bereitgestellt, um Sie beim Schreiben von C-Anwendungsprogrammen zu unterstützen, die die MQI verwenden.

Die Tabelle 469 auf Seite 245 enthält eine Übersicht über diese Headerdateien.

Tabelle 469. C-Headerdateien

Datei	**Inhalt**
CMQC	Funktionsprototypen, Datentypen und benannte Konstanten für das übergeordnete Message Queue Interface
CMQXC	Funktionsprototypen, Datentypen und benannte Konstanten für den Datenkonvertierungsexit
CMQEC	Funktionsprototypen, Datentypen und benannte Konstanten für das übergeordnete Message Queue Interface, den Datenkonvertierungsexit und die Struktur der Schnittstelleneingangspunkte (CMQEC beinhaltet CMQXC und CMQC).

Codieren Sie den Namen der Headerdatei zur Verbesserung der Portierbarkeit von Anwendungen in der Vorprozessoranweisung `#include` in Kleinbuchstaben:

```
#include "cmqec.h"
```

Funktionen
Sie müssen nicht alle Parameter angeben, die bei jedem Aufruf einer Funktion von Adresse übergeben werden.

- Übergeben Sie Parameter, die reine *Eingabeparameter* sind und den Typ MQHCONN, MQHOBJ oder MQLONG aufweisen, nach ihrem Wert.
- Übergeben Sie alle anderen Parameter nach ihrer Adresse.

Wird ein bestimmter Parameter nicht benötigt, verwenden Sie anstelle der Adresse der Parameterdaten beim Funktionsaufruf einen Nullzeiger als Parameter. Parameter, bei denen dies möglich ist, sind in den Aufrufbeschreibungen angegeben.

Als Wert der Funktion wird kein Parameter zurückgegeben; dies bedeutet in der Terminologie der Programmiersprache C, dass alle Funktionen `void` zurückgeben.

Die Attribute der Funktion werden durch die Makrovariable MQENTRY definiert. Der Wert dieser Makrovariablen hängt von der Umgebung ab.

Parameter mit einem nicht definierten Datentyp

Der Parameter *`Buffer`* der Funktionen MQGET, MQPUT und MQPUT1 weist einen nicht definierten Datentyp auf. Mit diesem Parameter werden die Nachrichtendaten einer Anwendung gesendet und empfangen.

Parameter dieser Art werden in den C-Beispielen als Arrays von MQBYTE dargestllt. Sie können die Parameter zwar auf diese Weise deklarieren, in der Regel ist es jedoch praktischer, sie als spezielle Struktur zu deklarieren, die den Aufbau der Daten in der Nachricht beschreibt. Deklarieren Sie den eigentlichen Funktionsparameter als typenlosen Zeiger und geben Sie beim Funktionsaufruf die Adresse beliebiger Daten als Parameter an.

Datentypen

Definieren Sie alle Datentypen mithilfe der Anweisung `typedef` der Programmiersprache C. Definieren Sie außerdem für jeden Datentyp den entsprechenden Zeigerdatentyp. Als Name des Zeigerdatentyps wird der Name des Elementar- oder Strukturdatentyps mit dem Präfix P verwendet, das auf einen Zeiger hinweist. Definieren Sie die Attribute des Zeigers mit der Makrovariablen MQPOINTER. Der Wert dieser Makrovariablen hängt von der Umgebung ab. Das folgende Beispiel veranschaulicht die Deklaration von Zeigerdatentypen:

```
#define MQPOINTER *               /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;    /* pointer to MQMD */
```

Binärzeichenfolgen bearbeiten
Deklarieren Sie Zeichenfolgen mit binären Daten als einen der MQBYTEn-Datentypen.

Verwenden Sie zum Kopieren, Vergleichen oder Festlegen von Feldern dieses Typs stets die Funktionen `memcpy`, `memcmp` bzw. `memset` der Programmiersprache C. Beispiel:

```
#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));
```

Verwenden Sie nicht die Zeichenfolgefunktionen `strcpy`, `strcmp`, `strncpy` oder `strncmp`, da diese bei Daten, die mit den MQBYTEn-Datentypen deklariert werden, nicht ordnungsgemäß funktionieren.

Zeichenfolgen bearbeiten
Wenn der Warteschlangenmanager Zeichendaten an die Anwendung zurückgibt, setzt der Warteschlangenmanager die Zeichendaten immer mit Leerzeichen auf die definierte Länge des Felds ein; der Warteschlangenmanager *Nicht* gibt die Zeichenfolgen mit dem Status 'null' zurück.

Verwenden Sie daher zum Kopieren, Vergleichen oder Verketten derartiger Zeichenfolgen stets die Zeichenfolgefunktionen `strncpy`, `strncmp` bzw. `strncat`.

Verwenden Sie nicht die Zeichenfolgefunktionen, für die die Zeichenfolge durch eine Null beendet werden muss (`strcpy`, `strcmp`, `strcat`). Verwenden Sie außerdem nicht die Funktion `strlen` , um die Länge der Zeichenfolge zu bestimmen. Verwenden Sie stattdessen die Funktion `sizeof` , um die Länge des Felds festzulegen.

Anfangswerte für Strukturen
Die Headerdateien definieren verschiedene Makrovariablen, die Sie verwenden können, um Anfangswerte für die MQ-Strukturen bereitzustellen, wenn Sie Instanzen dieser Strukturen deklarieren.

Die Namen dieser Makrovariablen haben das Format MQxxx_DEFAULT. Dabei steht MQxxx für den Namen der Struktur. Sie werden auf folgende Weise verwendet:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Für einige Zeichenfelder (beispielsweise die Felder *StrucId*, die in den meisten Strukturen vorkommen, oder das Feld *Format*, das im MQMD vorkommt) definiert das Message Queue Interface bestimmte Werte, die gültig sind. Für jeden der gültigen Werte werden *zwei* Makrovariablen zur Verfügung gestellt:

- Eine Makrovariable definiert den Wert als eine Zeichenfolge mit einer Länge, die ohne die implizierten Null-Entsprechungen genau der definierten Länge des Feldes entspricht. Für das Feld *Format* wird im MQMD beispielsweise die folgende Makrovariable bereitgestellt (¬ steht für ein Leerzeichen):

  ```
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Verwenden Sie diese Form bei den Funktionen memcpy und memcmp.
- Die andere Makrovariable definiert den Wert als Zeichenfeldgruppe; für den Namen dieser Makrovariablen wird der Name der Zeichenfolgeform mit dem Suffix _ARRAY verwendet. Beispiel:

  ```
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Verwenden Sie diese Form zur Initialisierung des Felds, wenn Sie eine Instanz der Struktur deklarieren, deren Werte von den Werten abweichen, die durch die Makrovariable MQMD_DEFAULT bereitgestellt werden. (Dies ist nicht immer erforderlich, da in einigen Umgebungen in beiden Situationen die Zeichenfolgeform des Werts verwendet werden kann. Sie können jedoch die Feldgruppenform für Deklarationen verwenden, da dies aus Gründen der Kompatibilität mit der Programmiersprache C++ erforderlich ist.)

Anfangswerte für dynamische Strukturen

Wenn eine variable Anzahl an Instanzen einer Struktur erforderlich ist, werden die Instanzen in der Regel im Hauptspeicher erstellt, der dynamisch mit der Funktion `calloc` oder `malloc` angefordert wird. Für die Initialisierung der Felder in solchen Strukturen wird folgendes Verfahren empfohlen:

1. Deklarieren Sie eine Instanz der Struktur unter Verwendung der entsprechenden MQxxx_DEFAULT-Makrovariablen für die Initialisierung der Struktur. Diese Instanz dient als Modell für andere Instanzen:

   ```
   MQMD Model = {MQMD_DEFAULT};  /* declare model instance */
   ```

   Durch die Codierung der Schlüsselwörter `static` bzw. `auto` in der Deklaration können Sie der Modellinstanz je nach Bedarf eine statische oder dynamische Laufzeit zuweisen.
2. Fordern Sie mit der Funktion `calloc` oder `malloc` einen Speicher für eine dynamische Instanz der Struktur an:

   ```
   PMQMD Instance;
   Instance = malloc(sizeof(MQMD));  /* get storage for dynamic instance */
   ```
3. Kopieren Sie die Modellinstanz mit der Funktion memcpy in die dynamische Instanz:

   ```
   memcpy(Instance,&Model,sizeof(MQMD));  /* initialize dynamic instance */
   ```

Verwendung in der Programmiersprache C++

Für die Programmiersprache C++ enthalten die Headerdateien die folgenden zusätzlichen Anweisungen, die nur einbezogen werden, wenn Sie einen C++-Compiler verwenden:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Notationskonventionen
Diese Informationen zeigen, wie Sie die Funktionen aufrufen und Parameter deklarieren.

In manchen Fällen handelt es sich bei den Parametern um Feldgruppen mit einer nicht festgelegten Größe. Bei diesen Parametern wird für die Darstellung einer numerischen Konstante der Buchstabe "n" in Kleinschreibung verwendet. Ersetzen Sie beim Codieren der Deklaration für einen solchen Parameter den Buchstaben "n" durch den erforderlichen numerischen Wert.

Programmierung in der Programmiersprache COBOL

Dieser Abschnitt enthält Informationen, die Sie bei der Verwendung der MQI aus der Programmiersprache COBOL unterstützen.

Kopierdateien

Die verschiedenen COPY-Dateien erleichtern Ihnen das Schreiben von Anwendungsprogrammen für COBOL, die MQI verwenden. Es gibt zwei Dateien, die benannte Konstanten enthalten, und zwei Dateien für jede der Strukturen.

Jede Struktur wird in zwei Formaten zur Verfügung gestellt: Das eine Format enthält Anfangswerte, während diese im anderen Format nicht enthalten sind:

- Verwenden Sie die Strukturen mit Anfangswerten im Arbeitsspeicherabschnitt eines COBOL-Programms; sie befinden sich in COPY-Dateien, deren Namen das Suffix V haben (V steht für "Values", also Werte).
- Verwenden Sie die Strukturen ohne Anfangswerte in der `LINKAGE SECTION` eines COBOL-Programms. Sie sind in COPY-Dateien mit Namen mit dem Suffix L (für Verknüpfung) enthalten.

Die Tabelle 470 auf Seite 248 enthält eine Übersicht über die COPY-Dateien. Die aufgelisteten Dateien sind nicht in allen Umgebungen verfügbar.

Tabelle 470. COPY-Dateien für COBOL

Datei (mit Anfangswerten)	**Datei (ohne Anfangswerte)**	**Inhalt**
CMQAIRV	CMQAIRL	Datensatz mit Authentifizierungsdaten
CMQBOV	CMQBOL	Struktur Startoptionen
CMQCIHV	CMQCIHL	Struktur des CICS-Informationsheaders
CMQCNOV	CMQCNOL	Optionsstruktur für die Verbindung
CMQDHV	CMQDHL	Struktur des Verteilungsheaders
CMQDLHV	CMQDLHL	Headerstruktur für nicht zustellbare Nachrichten
CMQDXPV	CMQDXPL	Parameterstruktur des Exits für Datenkonvertierung
CMQGMOV	CMQGMOL	Optionsstruktur für den Nachrichtenabruf
CMQIIHV	CMQIIHL	Struktur des IMS-Informationsheaders
CMQMDV	CMQMDL	Nachrichtendeskriptorstruktur
CMQMDEV	CMQMDEL	Struktur Nachrichtendeskriptorerweiterung
CMQMD1V	CMQMD1L	Struktur des Nachrichtendeskriptors der Version 1
CMQODV	CMQODL	Objektdeskriptorstruktur
CMQORV	CMQORL	Struktur des Objektdatensatzes
CMQPMOV	CMQPMOL	Optionsstruktur für die Nachrichteneinreihung
CMQRFHV	CMQRFHL	Struktur des Regel- und Formatierungsheaders
CMQRFH2V	CMQRFH2L	Struktur des Regel- und Formatierungsheaders der Version 2
CMQRMHV	CMQRMHL	Struktur Referenznachrichtenheader

Tabelle 470. COPY-Dateien für COBOL (Forts.)		
Datei (mit Anfangswerten)	**Datei (ohne Anfangswerte)**	**Inhalt**
CMQRRV	CMQRRL	Struktur des Antwortdatensatzes
CMQSCOV	CMQSCOL	SSL-Konfigurationsoptionen
CMQTMV	CMQTML	Auslösenachrichtenstruktur
CMQTMCV	CMQTMCL	Struktur der Auslösenachricht (Zeichenformat)
CMQTMC2V	CMQTMC2L	Struktur der Auslösenachricht (Zeichenformat) der Version 2
CMQWIHV	CMQWIHL	Auslastungsheaderstruktur
CMQXQHV	CMQXQHL	Headerstruktur Übertragungswarteschlange
CMQV	-	Benannte Konstanten für das übergeordnete Message Queue Interface
CMQXV	-	Benannte Konstanten für Datenkonvertierungsexit
CMQMD2V	CMQMD2L	Nachrichtendeskriptorstruktur Version 2

Strukturen

In der COPY-Datei beginnt jede Strukturdeklaration mit einem Element der Ebene 10; dadurch können Sie mehrere Instanzen der Struktur deklarieren, indem Sie die Deklaration der Ebene 01 codieren und anschließend eine COPY-Anweisung verwenden, mit der der Rest der Strukturdeklaration hineinkopiert wird. Fügen Sie mit dem Schlüsselwort `IN` einen Verweis auf die entsprechende Instanz hinzu:

```
* Declare two instances of MQMD
 01  MY-MQMD.
     COPY CMQMDV.
 01  MY-OTHER-MQMD.
     COPY CMQMDV.
*
* Set MSGTYPE field in MY-OTHER-MQMD
     MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-MQMD.
```

Richten Sie die Strukturen an den entsprechenden Grenzen aus. Wenn Sie mit einer COPY-Anweisung eine Struktur hinter einem Element einschließen, das kein Element der Ebene 01 ist, müssen Sie sicherstellen, dass die Struktur an der richtigen relativen Position ausgehend vom Anfang des Elements der Ebene 01 beginnt. Bei den meisten MQI-Strukturen ist eine 4-Byte-Ausrichtung erforderlich. Ausgenommen hiervon sind die Strukturen MQCNO, MQOD und MQPMO, die unter IBM i eine 16-Byte-Ausrichtung erfordern.

Im vorliegenden Abschnitt werden die Namen der Felder in Strukturen ohne Präfix angegeben. In der Programmiersprache COBOL wird als Präfix vor die Feldnamen der Name der Struktur, gefolgt von einem Bindestrich, gesetzt. Wenn der Strukturname jedoch mit einer Ziffer endet, was darauf hinweist, dass es sich bei der Struktur um eine zweite oder höhere Version der ursprünglichen Struktur handelt, ist die Ziffer nicht im Präfix enthalten. Die Feldnamen in der Programmiersprache COBOL werden in Großbuchstaben angezeigt (bei Bedarf können Sie jedoch auch Kleinbuchstaben oder eine Kombination aus Groß-/Kleinschreibung verwenden). Das Feld *MsgType*, das unter „MQMD - Nachrichtendeskriptor” auf Seite 399 beschrieben wird, wird beispielsweise in COBOL als MQMD-MSGTYPE dargestellt.

Da die Strukturen mit dem V-Suffix mit Anfangswerten für alle Felder deklariert werden, müssen Sie nur die Felder einstellen, in denen der bereitgestellte Anfangswert nicht dem gewünschten Wert entspricht.

Zeiger

Einige Strukturen müssen optionale Daten adressieren, die sich möglicherweise nicht an die Struktur angrenzen, wie z. B. die MQOR-und MQRR-Datensätze, die von der MQOD-Struktur adressiert werden.

Zur Adressierung dieser optionalen Daten enthalten die Strukturen Felder, die mit dem Zeigerdatentyp deklariert werden. COBOL unterstützt den Zeigerdatentyp jedoch nicht in allen Umgebungen. Daher

können die optionalen Daten auch mithilfe von Feldern adressiert werden, die die relative Position der Daten ausgehend vom Beginn der Struktur enthalten.

Wenn Sie eine Anwendung zwischen Umgebungen portieren möchten, vergewissern Sie sich, dass der Zeigerdatentyp in allen gewünschten Umgebungen verfügbar ist. Ist dies nicht der Fall, muss die Anwendung die optionalen Daten mithilfe der relativen Positionsfelder adressieren, statt hierfür die Zeigerfelder zu verwenden.

Deklarieren Sie in Umgebungen, in denen keine Zeiger unterstützt werden, die Zeigerfelder als Bytefolgen der entsprechenden Länge. Dabei ist der Anfangswert eine ausschließlich aus Nullen bestehende Bytefolge. Ändern Sie diesen Anfangswert nicht, wenn Sie die Abstandsfelder verwenden.

Benannte Konstanten

Im vorliegenden Abschnitt enthalten die Namen der Konstanten einen Unterstrich (_) als Bestandteil des Namens. In COBOL muss anstelle des Unterstrichs ein Bindestrich (-) verwendet werden.

Konstanten, die Zeichen-Zeichenfolgewerte haben, verwenden das einfache Anführungszeichen als Zeichenfolgebegrenzer ('). In einigen Umgebungen müssen Sie unter Umständen eine geeignete Compileroption angeben, um den Compiler dazu zu veranlassen, das einfache Anführungszeichen als Zeichenfolgebegrenzer anstelle des doppelten Anführungszeichen zu akzeptieren.

Die benannten Konstanten werden in den COPY-Dateien als Elemente der Ebene 10 deklariert. Zur Verwendung der Konstanten müssen Sie das Element der Ebene 01 explizit deklarieren und die Deklarationen der Konstanten anschließend mit der Anweisung COPY hineinkopieren:

```
* Declare a structure to hold the constants
 01  MY-MQ-CONSTANTS.
     COPY CMQV.
```

Beim obigen Verfahren belegen die Konstanten auch dann Speicher im Programm, wenn nicht auf sie verwiesen wird. Wenn Sie die Konstanten in viele separate Programme innerhalb derselben Ausführungseinheit einschließen, sind mehrere Kopien der Konstanten vorhanden, was zu einer unnötigen Belegung des Hauptspeichers führt. Dieser Umstand kann mit einem der folgenden Verfahren vermieden werden:

- Fügen Sie der Deklaration der Ebene 01 die Klausel GLOBAL hinzu:

  ```
  * Declare a global structure to hold the constants
   01  MY-MQ-CONSTANTS GLOBAL.
       COPY CMQV.
  ```

  Dies bewirkt, dass nur einer Gruppe von Konstanten innerhalb der Ausführungseinheit Speicher zugeordnet wird. Auf die Konstanten kann jedoch durch jedes beliebige Programm innerhalb der Ausführungseinheit verwiesen werden, nicht nur durch das Programm, das die Deklaration der Ebene 01 enthält.

  Anmerkung: Die Klausel GLOBAL wird nicht in allen Umgebungen unterstützt.
- Kopieren Sie manuell nur die Konstanten in die einzelnen Programme, auf die durch dieses Programm verwiesen wird. Kopieren Sie nicht alle Konstanten mit der Anweisung COPY in das Programm.

Notationskonventionen

An späterer Stelle in diesem Abschnitt wird erläutert, wie die Aufrufe aufgerufen und Parameter deklariert werden. In manchen Fällen handelt es sich bei den Parametern um Tabellen oder Zeichenfolgen mit einer nicht festgelegten Größe. Bei diesen Parametern wird für die Darstellung einer numerischen Konstante der Buchstabe "n" in Kleinschreibung verwendet. Ersetzen Sie beim Codieren der Deklaration für einen solchen Parameter den Buchstaben "n" durch den erforderlichen numerischen Wert.

Programmierung in der System/390-Assemblersprache

Dieser Abschnitt enthält hilfreiche Informationen für die Verwendung des Message Queue Interface mit der System/390-Assemblerprogrammiersprache.

Makros
Es werden verschiedene Makros bereitgestellt, die Ihnen helfen, Assembleranwendungsprogramme zu schreiben, die die MQI verwenden.

Es stehen zwei Makros für benannte Konstanten zur Verfügung, und ein Makro für jede der Strukturen. Die Tabelle 471 auf Seite 251 enthält eine Übersicht über diese Dateien.

Tabelle 471. Assemblermakros

Datei	**Inhalt**
CMQA	Benannte Konstanten (Gleichsetzungen) für das übergeordnete Message Queue Interface
CMQCIHA	Struktur des CICS-Informationsheaders
CMQCNOA	Optionsstruktur für die Verbindung
CMQDLHA	Headerstruktur für nicht zustellbare Nachrichten
CMQDXPA	Parameterstruktur des Exits für Datenkonvertierung
CMQGMOA	Optionsstruktur für den Nachrichtenabruf
CMQIIHA	Struktur des IMS-Informationsheaders
CMQMDA	Nachrichtendeskriptorstruktur
CMQMDEA	Struktur Nachrichtendeskriptorerweiterung
CMQODA	Objektdeskriptorstruktur
CMQPMOA	Optionsstruktur für die Nachrichteneinreihung
CMQRFHA	Struktur des Regel- und Formatierungsheaders
CMQRFH2A	Struktur des Regel- und Formatierungsheaders der Version 2
CMQRMHA	Struktur Referenznachrichtenheader
CMQTMA	Auslösenachrichtenstruktur
CMQTMC2A	Struktur der Auslösenachricht (Zeichenformat) der Version 2
CMQVERA	Versionssteuerung der Struktur
CMQWIHA	Auslastungsheaderstruktur
CMQXA	Benannte Konstanten für Datenkonvertierungsexit
CMQXPA	Parameterstruktur des API-Cross-Exits
CMQXQHA	Headerstruktur Übertragungswarteschlange

Strukturen
Die Strukturen werden durch Makros generiert, die verschiedene Parameter zur Steuerung der Aktion des Makros aufweisen. Diese Parameter werden in den folgenden Abschnitten beschrieben.

In regelmäßigen Abständen werden neue Versionen der MQ-Strukturen zur Verfügung gestellt. Die zusätzlichen Felder in einer neuen Version können dazu führen, dass eine Struktur, die zuvor kleiner als 256 Bytes war, diese Größe nun übersteigt. Schreiben Sie daher Assembleranweisungen, mit denen eine MQ-Struktur kopiert oder eine MQ-Struktur auf Nullwerte gesetzt werden soll, die sich für Strukturen eignen, die möglicherweise größer als 256 Bytes sind. Alternativ können Sie auch mit dem Makroparameter `DCLVER` oder mit dem Makro CMQVERA und dem Parameter `VERSION` eine bestimmte Version der Struktur deklarieren.

Strukturnamen angeben
Um mehr als eine Instanz einer Struktur zu deklarieren, setzt das Makro den Namen jedes Felds in der Struktur mit einer benutzerdefinierbaren Zeichenfolge und einem Unterstreichungszeichen fest.

Bei der verwendeten Zeichenfolge handelt es sich um den Kennsatz, der beim Aufruf des Makros angegeben wird. Erfolgt keine Angabe des Kennsatzes, wird für die Erstellung des Präfix der Name der Struktur verwendet:

```
* Declare two object descriptors
         CMQODA  ,        Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA  ,        Prefix used="MY_MQOD_"
```

Bei den Strukturdeklarationen in diesem Abschnitt wird das Standardpräfix verwendet.

Strukturformat angeben

Strukturdeklarationen können vom Makro in einem von zwei möglichen Formaten generiert werden. Dies wird durch den Parameter DSECT gesteuert:

DSECT=YES
Mit der Assembleranweisung DSECT wird ein neuer Datenabschnitt begonnen; die Strukturdefinition folgt unverzüglich auf die Anweisung DSECT. Der Kennsatz im Makroaufruf wird als Name des Datenabschnitts verwendet; wird kein Kennsatz angegeben, wird der Name der Struktur verwendet.

DSECT=NO
Mit den Assembleranweisungen DC wird die Struktur an der aktuellen Position in der Routine definiert. Die Felder werden mit Werten initialisiert, die durch die Codierung der relevanten Parameter im Makroaufruf angegeben werden können. Felder, für die im Makroaufruf keine Werte angegeben werden, werden mit den jeweiligen Standardwerten initialisiert.

Der Wert muss in Großbuchstaben angegeben werden. Falls für den Parameter DSECT keine Angabe erfolgt, wird der Wert DSECT=NO vorausgesetzt.

Strukturversion steuern
Standardmäßig deklarieren die Makros immer die neueste Version jeder Struktur.

Obwohl Sie den Makroparameter VERSION verwenden können, um einen Wert für das Feld *Version* in der Struktur anzugeben, definiert dieser Parameter den Anfangswert für das Feld *Version* und steuert nicht die Version der tatsächlich deklarierten Struktur. Zur Steuerung der deklarierten Strukturversion muss der Parameter DCLVER verwendet werden:

DCLVER=CURRENT
Die aktuelle (neueste) Version wird als Version deklariert.

DCLVER=SPECIFIED
Als deklarierte Version wird die Version übernommen, die durch den Parameter VERSION angegeben ist. Wenn Sie für den Parameter VERSION keinen Wert angeben, wird standardmäßig die Version 1 verwendet.

Falls Sie den Parameter VERSION angeben, muss der Wert eine selbstdefinierende numerische Konstante oder die benannte Konstante für die erforderliche Version sein (beispielsweise MQCNO_VERSION_3). Wenn Sie einen anderen Wert angeben, wird die Struktur deklariert, als ob DCLVER=CURRENT angegeben worden wäre. Dies gilt auch dann, wenn der Wert von VERSION in einen gültigen Wert aufgelöst wird.

Der Wert muss in Großbuchstaben angegeben werden. Wenn Sie den Parameter DCLVER übergehen, wird der verwendete Wert der globalen Makrovariablen MQDCLVER entnommen. Sie können diese Variable mit dem Makro CMQVERA festlegen.

Struktur deklarieren, die in eine andere Struktur eingebettet ist

Verwenden Sie den Parameter NESTED, um eine Struktur als Komponente einer anderen Struktur zu deklarieren:

NESTED=YES
Die Strukturdeklaration wird in eine andere Strukturdeklaration verschachtelt.

NESTED=NO
Die Strukturdeklaration wird nicht in eine andere Strukturdeklaration verschachtelt.

Der Wert muss in Großbuchstaben angegeben werden. Wenn Sie für den Parameter NESTED keinen Wert angeben, wird NESTED=NO vorausgesetzt.

Anfangswerte für Felder angeben
Geben Sie den Wert an, der zum Initialisieren eines Felds in einer Struktur verwendet werden soll, indem Sie den Namen dieses Felds (ohne das Präfix) als Parameter für den Makroaufruf codieren, der von dem erforderlichen Wert begleitet wird.

Wenn Sie beispielsweise eine Nachrichtendeskriptorstruktur deklarieren möchten, bei der das Feld *MsgType* mit MQMT_REQUEST und das Feld *ReplyToQ* mit der Zeichenfolge "MY_REPLY_TO_QUEUE" initialisiert wird, verwenden Sie folgenden Code:

```
MY_MQMD  CMQMDA  MSGTYPE=MQMT_REQUEST,                              X
               REPLYTOQ=MY_REPLY_TO_QUEUE
```

Wenn Sie eine benannte Konstante (Gleichsetzung) als Wert im Makroaufruf angeben, definieren Sie die benannte Konstante mit dem Makro CMQA. Schließen Sie Zeichenfolgewerte nicht in einfache Anführungszeichen ein.

Liste steuern

Mit dem Parameter LIST können Sie die Darstellung der Strukturdeklaration in der Assemblerliste steuern:

LIST=YES
Die Strukturdeklaration wird in der Assemblerliste angezeigt.

LIST=NO
Die Strukturdeklaration wird nicht in der Assemblerliste angezeigt.

Der Wert muss in Großbuchstaben angegeben werden. Wenn Sie für den Parameter LIST keinen Wert angeben, wird LIST=NO vorausgesetzt.

Makro CMQVERA

Dieses Makro ermöglicht die Festlegung des Standardwerts, der für den Parameter DCLVER in den Strukturmakros verwendet wird. Der durch CMQVERA angegebene Wert wird vom Strukturmakro nur verwendet, wenn der Parameter DCLVER im Aufruf des Strukturmakros übergangen wird. Der Standardwert wird durch die Codierung des Makros CMQVERA mit dem Parameter DCLVER festgelegt:

DCLVER=CURRENT
Die aktuelle (neueste) Version wird als Standardversion verwendet.

DCLVER=SPECIFIED
Die Standardversion wird auf die Version gesetzt, die durch den Parameter VERSION angegeben ist.

Die Angabe des Parameters DCLVER ist erforderlich. Der Wert muss in Großbuchstaben angegeben werden. Der durch CMQVERA festgelegte Wert wird bis zum nächsten Aufruf von CMQVERA oder bis zum Ende der Assemblierung als Standardwert verwendet. Wenn Sie keinen Wert für CMQVERA angeben, lautet die Standardeinstellung DCLVER=CURRENT.

Notationskonventionen

In nachfolgenden Abschnitten wird erläutert, wie die Aufrufe aufgerufen und Parameter deklariert werden. In manchen Fällen handelt es sich bei den Parametern um Feldgruppen oder Zeichenfolgen mit einer nicht festgelegten Größe, bei denen für die Darstellung einer numerischen Konstante der Buchstabe "n" in Kleinschreibung verwendet wird. Ersetzen Sie beim Codieren der Deklaration für einen solchen Parameter den Buchstaben "n" durch den erforderlichen numerischen Wert.

MQAIR - Datensätze für Authentifizierungsinformationen

Die MQAIR-Struktur steht für den Datensatz für Authentifizierungsinformationen.

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur.

Tabelle 472. Felder in MQAIR		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
AuthInfoType	Typ der Authentifizierungsdaten	AuthInfoType
AuthInfoConnName	Verbindungsname des LDAP-CRL-Servers	AuthInfoConnName
LDAPUserNamePtr	Adresse des LDAP-Benutzernamens	LDAPUserNamePtr
LDAPUserNameOffset	Relative Position des LDAP-Benutzernamens ab dem Anfang der MQSCO-Struktur	LDAPUserNameOffset
LDAPUserNameLength	Länge des LDAP-Benutzernamens	LDAPUserNameLength
LDAPPassword	Kennwort für den Zugriff auf den LDAP-Server	LDAPPassword
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQAIR_VERSION_2 ist.		
OCSPResponderURL	URL, unter der der OCSP-Responder kontaktiert werden kann	OCSPResponderURL

Übersicht über MQAIR

Die MQAIR-Struktur ermöglicht einer Anwendung, die als WebSphere MQ- MQI-Client ausgeführt wird, die Angabe von Informationen zu einem Authentifikator, der für die Clientverbindung verwendet werden soll. Die Struktur ist ein Eingabeparameter im MQCONNX-Anruf.

Verfügbarkeit: AIX, HP-UX, Solaris, Linux und Windows -Clients.

Zeichensatz und Codierung: Die Daten in MQAIR müssen den Zeichensatz und die Codierung des lokalen Warteschlangenmanagers aufweisen. Diese werden durch das Warteschlangenmanagerattribut **CodedCharSetId** und MQENC_NATIVE angegeben.

Felder für MQAIR

Die MQAIR-Struktur enthält die folgenden Felder; die Felder werden in **Alphabetische Reihenfolge**beschrieben:

AuthInfoConnName (MQCHAR264)
Dies ist entweder der Hostname oder die Netzwerkadresse eines Hosts, auf dem der LDAP-Server ausgeführt wird. Auf diese Angabe kann eine in Klammern gesetzte Anschlussnummer (optional) folgen. Die Standardportnummer ist 389.

Wenn der Wert kürzer als die Länge des Felds ist, beenden Sie den Wert mit einem Nullzeichen oder füllen Sie es mit Leerzeichen auf die Länge des Felds auf. Wenn der Wert nicht gültig ist, schlägt der Aufruf mit dem Ursachencode MQRC_AUTH_INFO_CONN_NAME_ERROR fehl.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_CONN_NAME_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.

AuthInfoType (MQLONG)
Dies ist der Typ der Authentifizierungsinformationen, die in dem Datensatz enthalten sind.

Als Wert kann einer der folgenden beiden Parameter verwendet werden:

MQAIT_CRL_LDAP
Überprüfung des Zertifikatswiderrufs mit dem LDAP-Server

MQAIT_OCSP
Überprüfung des Zertifikatswiderrufs mit OCSP

Wenn der Wert nicht gültig ist, schlägt der Aufruf mit dem Ursachencode MQRC_AUTH_INFO_TYPE_ERROR fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds lautet MQAIT_CRL_LDAP.

LDAPPassword (MQCHAR32)
Hier wird das Kennwort angegeben, das für den Zugriff auf den LDAP-CRL-Server erforderlich ist. Wenn der Wert kürzer ist als die Länge des Felds, beenden Sie den Wert mit einem Nullzeichen oder füllen Sie ihn mit Leerzeichen auf die Länge des Felds.

Wenn für den LDAP-Server kein Kennwort erforderlich ist oder Sie den LDAP-Benutzernamen übergehen, muss das Feld *LDAPPassword* null oder leer sein. Wird der LDAP-Benutzername übergangen und ist das Feld *LDAPPassword* nicht null oder leer, schlägt der Aufruf mit dem Ursachencode MQRC_LDAP_PASSWORD_ERROR fehl.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_LDAP_PASSWORD_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.

LDAPUserNameLength (MQLONG)

Hier wird die Bytelänge des LDAP-Benutzernamens angegeben, der durch das Feld *LDAPUserNamePtr* oder *LDAPUserNameOffset* adressiert wird. Der Wert muss zwischen 0 und MQ_DISTINGUISHED_NAME_LENGTH liegen. Wenn der Wert nicht gültig ist, schlägt der Aufruf mit dem Ursachencode MQRC_LDAP_USER_NAME_LENGTH_ERR fehl.

Wenn der einbezogene LDAP-Server keinen Benutzernamen erfordert, setzen Sie dieses Feld auf null.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

LDAPUserNameOffset (MQLONG)
Dies ist der Offset in Byte des LDAP-Benutzernamens ab dem Start der MQAIR-Struktur.

Der Offset kann positiv oder negativ sein. Das Feld wird ignoriert, wenn *LDAPUserNameLength* null ist.

Sie können entweder *LDAPUserNamePtr* oder *LDAPUserNameOffset* verwenden, um den LDAP-Benutzernamen einzugeben, nicht aber beide. Weitere Informationen finden Sie in der Beschreibung des Feldes *LDAPUserNamePtr*.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

LDAPUserNamePtr (PMQCHAR)
Dies ist der LDAP-Benutzername.

Hierbei handelt es sich um den definierten Namen des Benutzers, der versucht, auf den LDAP CRL-Server zuzugreifen. Ist der Wert kürzer als die durch *LDAPUserNameLength* angegebene Länge, beenden Sie den Wert mit einem Nullzeichen oder füllen Sie ihn mit Leerzeichen auf die Länge von *LDAPUserNameLength* auf. Das Feld wird ignoriert, wenn *LDAPUserNameLength* null ist.

Für die Bereitstellung des LDAP-Benutzernamens haben Sie zwei Möglichkeiten:

- Verwendung des Zeigerfelds *LDAPUserNamePtr*

  In diesem Fall kann die Anwendung eine Zeichenfolge deklarieren, die von der MQAIR-Struktur getrennt ist, und *LDAPUserNamePtr* auf die Adresse der Zeichenfolge setzen.

  Es kann sinnvoll sein, *LDAPUserNamePtr* bei Programmiersprachen zu verwenden, die den Zeigerdatentyp auf eine Weise unterstützen, die auf verschiedene Umgebungen übertragbar ist (beispielsweise bei der Programmiersprache C).
- Verwendung des relativen Positionsfelds *LDAPUserNameOffset*

In diesem Fall muss die Anwendung eine zusammengesetzte Struktur deklarieren, die die MQSCO-Struktur enthält, der die Feldgruppe der MQAIR-Datensätze und die LDAP-Zeichenfolgen für Benutzernamen folgen. Sie muss *LDAPUserNameOffset* auf die relative Position der entsprechenden Namensfolge ab Beginn der MQAIR-Struktur setzen. Stellen Sie sicher, dass dieser Wert korrekt ist und dass es sich um einen Wert handelt, der in MQLONG aufgenommen werden kann (die restriktivste Programmiersprache ist COBOL, bei der der gültige Bereich von -999 999 999 bis +999 999 999 reicht).

Es kann sinnvoll sein, *LDAPUserNameOffset* bei Programmiersprachen zu verwenden, die den Zeigerdatentyp nicht unterstützen oder den Zeigerdatentyp auf eine Weise implementieren, die möglicherweise nicht auf verschiedene Umgebungen übertragbar ist (beispielsweise bei der Programmiersprache COBOL).

Ungeachtet des gewählten Verfahrens darf nur entweder *LDAPUserNamePtr* oder *LDAPUserNameOffset* verwendet werden; haben beide Felder einen Wert ungleich null, schlägt der Aufruf mit dem Ursachencode MQRC_LDAP_USER_NAME_ERROR fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge.

Anmerkung: Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

OCSPResponderURL (MQCHAR256)
Für eine MQAIR-Struktur, die Verbindungsdetails für einen OCSP-Responder darstellt, enthält dieses Feld die URL, unter der eine Verbindung zum Responder hergestellt werden kann.

Der Wert dieses Feldes ist eine HTTP-URL. Dieses Feld hat Vorrang vor einer URL in einer AIA-Zertifikatserweiterung (AIA - AuthorityInfoAccess).

Der Wert wird ignoriert, es sei denn, die beiden folgenden Aussagen sind wahr:

- Die MQAIR-Struktur weist die Version 2 oder eine spätere Version auf (das Feld "Version" ist auf den Wert MQAIR_VERSION_2 oder einen höheren Wert gesetzt).
- Das Feld "AuthInfoType" ist auf MQAIT_OCSP gesetzt.

Wenn das Feld keine HTTP-URL im korrekten Format enthält (und nicht ignoriert wird), dann schlägt der MQCONNX-Aufruf fehl und das System gibt den Ursachencode MQRC_OCSP_URL_ERROR aus.

Bei diesem Feld muss die Groß-/Kleinschreibung beachtet werden. Es muss mit der Zeichenfolge http:// in Kleinschreibung beginnen. Der Rest der URL kann abhängig von der OCSP-Serverimplementierung die Groß-/Kleinschreibung beachten.

Bei diesem Feld findet keine Datenkonvertierung statt.

StrucId (MQCHAR4)

Folgende Werte sind möglich:

MQAIR_STRUC_ID
ID für den Datensatz mit Authentifizierungsinformationen.

Für die Programmiersprache C ist auch die Konstante MQAIR_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQAIR_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds lautet MQAIR_STRUC_ID.

Version (MQLONG)
Die Versionsnummer der MQAIR-Struktur.

Folgende Werte sind möglich:

MQAIR_VERSION_1
Der Authentifizierungsdatensatz der Version 1.

MQAIR_VERSION_2
Der Authentifizierungsdatensatz der Version 2.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQAIR_CURRENT_VERSION
Aktuelle Version des Datensatzes mit Authentifizierungsinformationen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds lautet MQAIR_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQAIR

Tabelle 473. Anfangswerte der Felder in MQAIR

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQAIR_STRUC_ID	'AIR¬'
Version	MQAIR_VERSION_1	1
AuthInfoType	MQAIT_CRL_LDAP	1
AuthInfoConnName	--	Nullzeichenfolge oder Leerzeichen.
LDAPUserNamePtr	--	Nullzeiger oder Null Byte
LDAPUserNameOffset	--	0
LDAPUserNameLength	--	0
LDAPPassword	--	Nullzeichenfolge oder Leerzeichen.
OCSPResponderURL	--	Nullzeichenfolge oder Leerzeichen.

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQAIR_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQLONG     AuthInfoType;        /* Type of authentication
                                     information */
  MQCHAR264  AuthInfoConnName;    /* Connection name of CRL LDAP
                                     server */
  PMQCHAR    LDAPUserNamePtr;     /* Address of LDAP user name */
  MQLONG     LDAPUserNameOffset;  /* Offset of LDAP user name from start
                                     of MQAIR structure */
  MQLONG     LDAPUserNameLength;  /* Length of LDAP user name */
  MQCHAR32   LDAPPassword;        /* Password to access LDAP server */
  MQCHAR256  OCSPResponderURL;    /* URL of OCSP responder */

};
```

COBOL-Delaration

```
**   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).
```

Deklaration in Visual Basic

```
Type MQAIR
  StrucId            As String*4   'Structure identifier'
  Version            As Long       'Structure version number'
  AuthInfoType       As Long       'Type of authentication information'
  AuthInfoConnName   As String*264 'Connection name of CRL LDAP server'
  LDAPUserNamePtr    As MQPTR      'Address of LDAP user name'
  LDAPUserNameOffset As Long       'Offset of LDAP user name from start'
                                   'of MQAIR structure'
  LDAPUserNameLength As Long       'Length of LDAP user name'
  LDAPPassword       As String*32  'Password to access LDAP server'
End Type
```

MQBMHO - Puffer-zu-Nachrichtenhandle-Optionen

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. MQBMHO-Struktur-Puffer-zu-Nachrichtenhandle-Optionen

Tabelle 474. Felder in MQBMHO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQBMHO	Options

Überblick über MQBMHO

Verfügbarkeit: Alle. Puffer für die Struktur der Optionen für die Nachrichtenkennung – Übersicht

Zweck: Mit der MQBMHO-Struktur können Anwendungen Optionen festlegen, die steuern, wie Nachrichtenkennungen aus Puffern erzeugt werden. Bei der Struktur handelt es sich um einen Eingabeparameter im MQBUFMH-Aufruf.

Zeichensatz und Codierung: Die Daten in MQBMHO müssen im Zeichensatz und in der Codierung der Anwendung (MQENC_NATIVE) enthalten sein.

Felder für MQBMHO

Optionsstruktur Puffer für Nachrichtenkennung - Felder

Die MQBMHO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)
Pufferstruktur für Nachrichtenkennung - Feld Options

Folgende Werte sind möglich:

MQBMHO_DELETE_PROPERTIES
Eigenschaften, die zur Nachrichtenkennung hinzugefügt werden, werden aus dem Puffer gelöscht. Schlägt der Aufruf fehl, werden keine Eigenschaften gelöscht.

Standardoptionen: Verwenden Sie folgende Option, wenn sie die beschriebene Option benötigen:

MQBMHO_NONE
Keine Optionen angegeben.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQBMHO_DELETE_PROPERTIES.

StrucId (MQCHAR4)
Pufferstruktur für Nachrichtenkennung - Feld StrucId

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQBMHO_STRUC_ID
ID der Puffer-zu-Nachrichtenkennung-Struktur.

Für die Programmiersprache C wird auch die Konstante MQBMHO_STRUC_ID_ARRAY definiert. Diese hat denselben Wert wie MQBMHO_STRUC_ID, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQBMHO_STRUC_ID.

Version (MQLONG)
Pufferstruktur für Nachrichtenkennung - Feld Version

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQBMHO_VERSION_1
Versionsnummer der Puffer-zu-Nachrichtenkennung-Struktur.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQBMHO_CURRENT_VERSION
Aktuelle Version der Puffer-zu-Nachrichtenkennung-Struktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQBMHO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQBMHO
Pufferstruktur für Nachrichtenkennung - Anfangswerte

Tabelle 475. Anfangswerte der Felder in MQBMHO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQBMHO_STRUC_ID	'BMHO'
Version	MQBMHO_VERSION_1	1
Options	MQBMHO_NONE	0

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQBMHO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQBMHO MyBMHO = {MQBMHO_DEFAULT};
```

Deklaration in Programmiersprache C
Pufferstruktur für Nachrichtenkennung - Deklaration für Programmiersprache C

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQBUFMH */
};
```

COBOL-Delaration
Pufferstruktur für Nachrichtenkennung - Deklaration für Programmiersprache COBOL

```
**   MQBMHO structure
  10 MQBMHO.
**    Structure identifier
   15 MQBMHO-STRUCID               PIC X(4).
**    Structure version number
   15 MQBMHO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQBUFMH
   15 MQBMHO-OPTIONS               PIC S9(9) BINARY.
```

Deklaration in PL/I
Pufferstruktur für Nachrichtenkennung - Deklaration für Programmiersprache PL/I

```
Dcl
 1 MQBMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQBUFMH */
```

Deklaration in High Level Assembler
Pufferstruktur für Nachrichtenkennung - Deklaration für Programmiersprache Assembler

```
MQBMHO                    DSECT
MQBMHO_STRUCID            DS   CL4  Structure identifier
MQBMHO_VERSION            DS   F    Structure version number
MQBMHO_OPTIONS            DS   F    Options that control the
*                                    action of MQBUFMH
MQBMHO_LENGTH             EQU  *-MQBMHO
MQBMHO_AREA               DS   CL(MQBMHO_LENGTH)
```

MQBO - Startoptionen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 476. Felder in MQBO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQBEGIN	Options

Übersicht über MQBO

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows; nicht verfügbar für WebSphere MQ MQI-Clients.

Zweck: Die MQBO-Struktur ermöglicht der Anwendung die Angabe von Optionen im Zusammenhang mit der Erstellung einer Arbeitseinheit. Bei der Struktur handelt es sich um einen Ein-/Ausgabeparameter im MQBEGIN-Aufruf.

Zeichensatz und Codierung: Die Daten in MQBO müssen den Zeichensatz haben, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und die Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE festgelegt wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQBO

Die MQBO-Struktur enthält die folgenden Felder, die in **alphabetischer Reihenfolge** beschrieben werden:

Optionen (MQLONG)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert lautet MQBO_NONE.

Folgende Werte sind möglich:

MQBO_NONE
Keine Optionen angegeben.

StrucId (MQCHAR4)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert lautet MQBO_STRUC_ID.

Folgende Werte sind möglich:

MQBO_STRUC_ID
Die ID für die Struktur der Startoptionen.

Für die Programmiersprache C ist auch die Konstante MQBO_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQBO_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Version (MQLONG)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert lautet MQBO_VERSION_1.

Folgende Werte sind möglich:

MQBO_VERSION_1
Die Versionsnummer für die Struktur der Startoptionen.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQBO_CURRENT_VERSION
Die aktuelle Version der Struktur der Startoptionen.

Anfangswerte und Sprachendeklarationen für MQBO

Tabelle 477. Anfangswerte der Felder in MQBO für MQBO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQBO_STRUC_ID	'BO¬¬'
Version	MQBO_VERSION_1	1
Options	MQBO_NONE	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQBO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
  MQCHAR4  StrucId;  /* Structure identifier */
  MQLONG   Version;  /* Structure version number */
  MQLONG   Options;  /* Options that control the action of MQBEGIN */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQBO based,
  3 StrucId char(4),       /* Structure identifier */
  3 Version fixed bin(31), /* Structure version number */
  3 Options fixed bin(31); /* Options that control the action of
                              MQBEGIN */
```

Deklaration in Visual Basic

```
Type MQBO
  StrucId As String*4 'Structure identifier'
  Version As Long     'Structure version number'
  Options As Long     'Options that control the action of MQBEGIN'
End Type
```

MQCBC – Callback-Kontext

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. Struktur, die die Callback-Routine beschreibt.

Tabelle 478. Felder in MQCBC

Feld	Beschreibung	Thema
StrucID	Struktur-ID	StrucID
Version	Strukturversionsnummer	Version
CallType	Grund für Aufrufen der Funktion	CallType
Hobj	Objektkennung	Hobj
CallbackArea	Feld für zu verwendende Callback-Funktion	CallbackArea
ConnectionArea	Feld für zu verwendende Callback-Funktion	ConnectionArea
CompCode	Beendigungscode	CompCode
Reason	Ursachencode	Reason
State	Angabe des Status des aktuellen Nutzers	Staat
DataLength	Nachrichtenlänge	DataLength
BufferLength	Länge des Nachrichtenpuffers in Bytes	BufferLength

Tabelle 478. Felder in MQCBC (Forts.)		
Feld	**Beschreibung**	**Thema**
Flags	Allgemeine Flags	Flags
Anmerkung: Das übrige Feld wird ignoriert, wenn "Version" kleiner als MQCBC_VERSION_2 ist.		
ReconnectDelay	Anzahl der Millisekunden vor dem Versuch einer Verbindungswiederholung	ReconnectDelay

Überblick für MQCBC

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSsowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQCBC-Struktur wird zum Angeben von Kontextinformationen verwendet, die an eine Callback-Funktion übergeben werden.

Die Struktur ist ein Ein-/Ausgabeparameter im Aufruf an eine Nachrichtenkonsumentenroutine.

Version: Die aktuelle Version von MQCBC ist MQCBC_VERSION_2.

Zeichensatz und Codierung: Daten in MQCBC müssen im Zeichensatz vorliegen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird, sowie in der Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angegeben wird. Wenn die Anwendung jedoch als MQ MQI-Client ausgeführt wird, ist die Struktur im Zeichensatz und in der Codierung des Clients vorhanden.

Felder für MQCBC

Alphabetische Liste der Felder für die MQCBC-Struktur.

Die MQCBC-Struktur enthält die folgenden Felder, die in alphabetischer Reihenfolge beschrieben werden:

BufferLength (MQLONG)
Dieses Feld ist die Länge des Nachrichtenpuffers, der an diese Funktion übergeben wurde, in Bytes.

Der Puffer kann sowohl größer sein als der für den Konsumenten definierte MaxMsgLength-Wert als auch größer als der ReturnedLength-Wert in der MQGMO-Struktur.

Die tatsächliche Nachrichtenlänge wird im Feld DataLength angegeben.

Die Anwendung kann den gesamten Puffer während der Dauer der Callback-Funktion für ihre eigenen Zwecke verwenden.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion, das keine Relevanz für eine Funktion der Ausnahmebehandlungsroutine hat.

CallbackArea (MQPTR)
Dieses Feld ist für die Verwendung durch die Callback-Funktion verfügbar.

Der Warteschlangenmanager trifft keine Entscheidungen auf der Basis des Inhalts dieses Felds und er wird unverändert aus dem Feld „CallbackArea (MQPTR)" auf Seite 271 in der MQCBD-Struktur übergeben, bei dem es sich um einen Parameter im MQCB-Aufruf handelt, mit dem die Callback-Funktion definiert wird.

Änderungen an *CallbackArea* werden über die Aufrufe der Callback-Funktion für eine *HObj*hinweg beibehalten. Dieses Feld wird nicht gemeinsam mit Callback-Funktionen für andere Kennungen verwendet.

Dies ist ein Ein-/Ausgabefeld für die Callback-Funktion. Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

CallType (MQLONG)
Ein Feld, das Informationen darüber enthält, warum diese Funktion aufgerufen wurde; folgende Optionen sind definiert.

Aufruftypen für die Nachrichtenübermittlung: Diese Aufruftypen enthalten Informationen über eine Nachricht. Die Parameter *DataLength* und *BufferLength* sind für diese Aufruftypen gültig.

MQCBCT_MSG_REMOVED
Die Nachrichtenkonsumentenfunktion wurde mit einer Nachricht aufgerufen, die aus der Objektkennung gelöscht wurde.

Wenn der Wert von *CompCode* MQCC_WARNING ist, ist der Wert des Feldes *Reason* MQRC_TRUNCATED_MSG_ACCEPTED oder einer der Codes, der ein Datenkonvertierungsproblem anzeigt.

MQCBCT_MSG_NOT_REMOVED
Die Nachrichtenkonsumentenfunktion wurde mit einer Nachricht aufgerufen, die noch nicht unwiederbringlich aus der Objektkennung entfernt wurde. Die Nachricht kann unter Verwendung des *MsgToken* aus der Objektkennung gelöscht werden.

Die Nachricht wurde möglicherweise aus einem der folgenden Gründe nicht entfernt:

- Die MQGMO-Optionen haben eine Suchoperation angefordert (MQGMO_BROWSE_*).
- Die Nachricht ist größer als der verfügbare Puffer und die MQGMO-Optionen geben nicht MQGMO_ACCEPT_TRUNCATED_MSG an.

Wenn der Wert von *CompCode* MQCC_WARNING ist, ist der Wert des Felds *Reason* MQRC_TRUNCATED_MSG_FAILED oder einer der Codes, die ein Datenkonvertierungsproblem angeben.

Aufruftypen für die Callback-Steuerung: Diese Aufruftypen enthalten Informationen zur Callback-Steuerung und keine Einzelheiten über eine Nachricht. Diese Aufruftypen werden mit Options in der MQCBD-Struktur angefordert.

Die Parameter *DataLength* und *BufferLength* sind für diese Aufruftypen nicht gültig.

MQCBCT_REGISTER_CALL
Dieser Aufruftyp ermöglicht der Callback-Funktion die Durchführung einer gewissen Erstkonfiguration.

Die Callback-Funktion wird unmittelbar nach der Registrierung des Rückrufs aufgerufen, d. h. bei der Rückgabe von einem MQCB-Aufruf mit einem Wert für das Feld *Operation* von MQOP_REGISTER.

Dieser Aufruftyp wird sowohl für Nachrichtenkonsumenten als auch für Ereignishandler verwendet.

Wenn der Typ angefordert wird, ist dies der erste Aufruf der Callback-Funktion.

Der Wert des Felds *Reason* ist MQRC_NONE.

MQCBCT_START_CALL
Der Zweck dieses Aufruftyps besteht darin, der Callback-Funktion zu ermöglichen, eine Konfiguration beim Start vorzunehmen, z. B. Ressourcen wiederherzustellen, die bereinigt wurden, als die Funktion zuvor gestoppt wurde.

Die Callback-Funktion wird beim Starten der Verbindung mit MQOP_START oder MQOP_START_WAIT aufgerufen.

Wenn eine Callback-Funktion in einer anderen Callback-Funktion registriert ist, wird dieser Aufruftyp aufgerufen, wenn der Callback zurückgegeben wird.

Dieser Aufruftyp wird nur für Nachrichtenkonsumenten verwendet.

Der Wert des Felds *Reason* ist MQRC_NONE.

MQCBCT_STOP_CALL
Dieser Aufruftyp ermöglicht der Callback-Funktion die Durchführung einer gewissen Bereinigung, wenn Sie vorübergehend angehalten wird, wie etwa die Bereinigung zusätzlicher Ressourcen, die während der Verarbeitung von Nachrichten angefordert wurden.

Die Callback-Funktion wird aufgerufen, wenn ein MQCTL-Aufruf mit einem Wert für das Feld *Operation* von MQOP_STOP ausgegeben wird.

Dieser Aufruftyp wird nur für Nachrichtenkonsumenten verwendet.

Der Wert des Felds *Reason* gibt den Grund für das Stoppen an.

MQCBCT_DEREGISTER_CALL
Dieser Aufruftyp ermöglicht der Callback-Funktion die Durchführung einer abschließenden Bereinigung am Ende der Verarbeitung. Die Callback-Funktion wird in folgenden Fällen aufgerufen:

- Die Registrierung der Callback-Funktion mit einem MQCB-Aufruf mit MQOP_DEREGISTER zurückgenommen wird.
- Die Warteschlange wird geschlossen, was eine implizite Aufhebung der Registrierung zur Folge hat. In diesem Fall wird der Callback-Funktion MQHO_UNUSABLE_HOBJ als Objektkennung übergeben.
- MQDISC-Aufruf wird beendet, was zu einem implizierten Schließvorgang und somit zu einem Zurücknehmen der Registrierung führt. In diesem Fall wird die Verbindung nicht sofort unterbrochen und laufende Transaktionen werden noch nicht ausgeführt.

Wird eine dieser Aktionen innerhalb der Callback-Funktion ausgeführt, wird die entsprechende Aktion ausgeführt, wenn der Callback zurückkehrt.

Dieser Aufruftyp wird sowohl für Nachrichtenkonsumenten als auch für Ereignishandler verwendet.

Soweit angefordert, ist dies der letzte Aufruf der Callback-Funktion.

Der Wert des Felds *Reason* gibt den Grund für das Stoppen an.

MQCBCT_EVENT_CALL
Ereignishandler-Funktion

Die Ereignishandler-Funktion wird ohne Nachricht aufgerufen, wenn der Warteschlangenmanager oder die Verbindung stoppt oder in den Wartemodus übergeht.

Dieser Aufruf kann verwendet werden, um entsprechende Aktionen für alle Callback-Funktionen auszuführen.

Nachrichtenkonsumentenfunktion

Die Nachrichtenkonsumentenfunktion wurde ohne Nachricht aufgerufen, wenn ein Fehler (*CompCode* = MQCC_FAILED) festgestellt wurde, der für die Objektkennung spezifisch ist. Beispiel: *Reason* code = MQRC_GET_INHIBIT.

Der Wert des Felds *Reason* wird festgelegt, um den Grund für den Aufruf anzugeben.

MQCBCT_MC_EVENT_CALL

Die Ereignishandler-Funktion wurde für Multicasting-Ereignisse aufgerufen. Anstelle "normaler" WebSphere MQ-Ereignisse werden WebSphere MQ Multicasting-Ereignisse an den Ereignishandler gesendet.

Weitere Informationen zu MQCBCT_MC_EVENT_CALL finden Sie im Abschnitt Multicasting-Ausnahmeberichterstellung.

CompCode (MQLONG)
Dieses Feld ist der Beendigungscode. Dieser gibt an, ob beim Verarbeiten der Nachricht Probleme aufgetreten sind.

Folgende Werte sind möglich:

MQCC_OK
Erfolgreiche Ausführung.

MQCC_WARNING
Warnung (teilweise Ausführung).

MQCC_FAILED
Aufruf fehlgeschlagen.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist MQCC_OK.

ConnectionArea (MQPTR)
Dieses Feld ist für die Verwendung durch die Callback-Funktion verfügbar.

Der Warteschlangenmanager trifft keine Entscheidungen auf der Basis des Inhalts dieses Felds und es wird unverändert aus dem Feld „ConnectionArea (MQPTR)" auf Seite 320 in der MQCTLO-Struktur übergeben, bei dem es sich um einen Parameter im MQCTL-Aufruf handelt, der zur Steuerung der Callback-Funktion verwendet wird.

Alle von den Callback-Funktionen an diesem Feld vorgenommenen Änderungen bleiben für alle Aufrufe der Callback-Funktion erhalten. Dieser Bereich kann für die Weitergabe von Informationen verwendet werden, die von allen Callback-Funktionen gemeinsam genutzt werden sollen. Im Gegensatz zu *CallbackArea* ist dieser Bereich für alle Callbacks für eine Verbindungskennung einheitlich.

Dies ist ein Ein-/Ausgabefeld. Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

DataLength (MQLONG)
Gibt die Länge der in der Nachricht enthaltenen Anwendungsdaten in Byte an. Eine Länge von null bedeutet, dass die Nachricht keine Anwendungsdaten enthält.

Das Feld DataLength enthält die Länge der Nachricht, aber nicht notwendigerweise die Länge der an den Konsumenten übergebenen Nachrichtendaten. Möglicherweise wurde die Nachricht abgeschnitten. Verwenden Sie das Feld ReturnedLength in der MQGMO-Struktur, um zu ermitteln, wie viele Daten tatsächlich an den Konsumenten übergeben wurden.

Wenn der Ursachencode angibt, dass die Nachricht abgeschnitten wurde, können Sie mit dem Feld DataLength ermitteln, wie groß die tatsächliche Nachricht ist. Dadurch können Sie die Größe des Puffers ermitteln, der zum Unterbringen der Nachrichtendaten erforderlich ist, und anschließend einen MQCB-Aufruf ausgeben, um die MaxMsgLength mit einem entsprechenden Wert zu aktualisieren.

Wenn die Option MQGMO_CONVERT angegeben wird, könnte die konvertierte Nachricht größer sein als der für DataLength zurückgegebene Wert. In diesen Fällen muss die Anwendung wahrscheinlich einen MQCB-Aufruf ausgeben, um die MaxMsgLength so zu aktualisieren, dass sie größer ist als der vom Warteschlangenmanager für DataLength zurückgegebene Wert.

Um Probleme beim Abschneiden von Nachrichten zu vermeiden, geben Sie MaxMsgLength als MQCBD_FULL_MSG_LENGTH an. Dies bewirkt, dass der Warteschlangenmanager einen Puffer für die gesamte Nachrichtenlänge nach der Datenkonvertierung zuweist. Aber auch wenn diese Option angegeben ist, besteht die Möglichkeit, dass für die korrekte Verarbeitung der Anforderung kein ausreichender Speicherplatz verfügbar ist. Der zurückgegebene Ursachencode muss von den Anwendungen immer überprüft werden. Wenn es beispielsweise nicht möglich ist, genügend Speicher zum Konvertieren der Nachricht zu reservieren, werden die Nachrichten unkonvertiert an die Anwendung zurückgegeben.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion; sie ist für eine Ereignishandler-Funktion ohne Relevanz.

Flags (MQLONG)
Flags, die Informationen über diesen Konsumenten enthalten.

Folgende Option ist definiert:

MQCBCF_READA_BUFFER_EMPTY
Dieses Flag kann zurückgegeben werden, wenn ein vorheriger MQCLOSE-Aufruf mit der Option MQCO_QUIESCE mit dem Ursachencode MQRC_READ_AHEAD_MSGS fehlgeschlagen ist.

Dieser Code weist daraufhin, dass die letzte Vorauslesenachricht zurückgegeben wird und der Puffer jetzt leer ist. Wenn die Anwendung einen weiteren MQCLOSE-Aufruf mit der Option MQCO_QUIESCE ausgibt, wird dieser erfolgreich ausgeführt.

Beachten Sie, dass nicht gewährleistet ist, dass eine Nachricht mit diesem Flag zur Anwendung übermittelt wird, da der Vorauslesepuffer noch Nachrichten enthalten kann, die mit den aktuellen

Auswahlkriterien nicht übereinstimmen. In diesem Fall wird die Konsumentenfunktion mit dem Ursachencode MQRC_HOBJ_QUIESCED aufgerufen.

Wenn der Vorauslesepuffer leer ist, wird der Konsument mit dem Flag MQCBCF_READA_BUFFER_EMPTY und dem Ursachencode MQRC_HOBJ_QUIESCED_NO_MSGS aufgerufen.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion; sie ist für eine Ereignishandler-Funktion ohne Relevanz.

Hobj (MQHOBJ)
Dies ist die Objektkennung für Aufrufe an den Nachrichtenkonsumenten.

Für einen Ereignishandler ist dieser Wert MQHO_NONE

Die Anwendung kann diese Kennung und den Nachrichtentoken im Block "Nachrichtenabrufoptionen" verwenden, um die Nachricht abzurufen, wenn eine Nachricht nicht aus der Warteschlange entfernt wurde.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQHO_UNUSABLE_HOBJ

Ursache (MQLONG)
Dies ist der Ursachencode, der *CompCode*qualifiziert.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist MQRC_NONE.

State (MQLONG)
Eine Angabe zum Status des aktuellen Konsumenten. Dieses Feld ist für eine Anwendung von dem meisten Wert, wenn ein Ursachencode ungleich null an die Konsumentenfunktion übergeben wird.

Sie können dieses Feld zum Vereinfachen der Anwendungsprogrammierung verwenden, da Sie das Verhalten nicht für jeden Ursachencode codieren müssen.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist MQCS_NONE

Staat	**Warteschlangenmanageraktion**	**Wert der Konstanten**
MQCS_NONE Dieser Ursachencode steht für einen normalen Aufruf ohne zusätzliche Informationen zur Ursache.	Keine; es handelt sich um den normalen Ablauf.	0
MQCS_SUSPENDED_TEMPORARY Diese Ursachencodes stehen für temporäre Bedingungen.	Die Callback-Routine wird zum Abrufen der Bedingung aufgerufen und anschließend ausgesetzt. Nach einem bestimmten Zeitraum versucht das System möglicherweise, die Operation erneut auszuführen. Dies könnte dazu führen, dass dieselbe Bedingung erneut auftritt.	1
MQCS_SUSPENDED_USER_ACTION Diese Ursachencodes stellen Bedingungen dar, unter denen der Callback eine Aktion ausführen muss, um die Bedingung zu beheben.	Der Konsument wird ausgesetzt und die Callback-Routine wird zum Abrufen der Bedingung aufgerufen. Falls möglich, sollte die Callback-Routine die Bedingung beheben und die Verbindung fortsetzen (RESUME) oder schließen.	2

Staat	Warteschlangenmanageraktion	Wert der Konstanten
MQCS_SUSPENDED Diese Ursachencodes stehen für Störungen, die weitere Nachrichten-Callbacks verhindern.	Der Warteschlangenmanager setzt die Callback-Funktion automatisch aus. Wird die Callback-Funktion wiederaufgenommen, wird wahrscheinlich derselbe Ursachencode erneut zurückgegeben.	3
MQCS_STOPPED Diese Ursachencodes stehen für das Ende der Nachrichtenverarbeitung.	Sie werden an die Ausnahmebehandlungsroutine und an Callbacks übergeben, die MQCBDO_STOP_CALL angegeben haben. Es können keine weiteren Nachrichten verarbeitet werden.	4

StrucId (MQCHAR4)
Der Wert in diesem Feld ist die Struktur-ID.

Folgende Werte sind möglich:

MQCBC_STRUC_ID
ID für die Callback-Kontextstruktur.

Für die Programmiersprache C ist auch die Konstante MQCBC_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQCBC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCBC_STRUC_ID.

Version (MQLONG)
Der Wert in diesem Feld ist die Strukturversionsnummer.

Folgende Werte sind möglich:

MQCBC_VERSION_1
Callback-Kontextstruktur der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCBC_CURRENT_VERSION
Aktuelle Version der Callback-Kontextstruktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCBC_VERSION_1.

Der Callback-Funktion wird immer die neueste Version der Struktur übergeben.

ReconnectDelay (MQLONG)
ReconnectDelay gibt an, wie lange der Warteschlangenmanager wartet, bevor er versucht, die Verbindung wiederherzustellen. Das Feld kann von einem Ereignishandler geändert werden, um die Verzögerung zu ändern oder die Verbindungswiederherstellung zu stoppen.

Verwenden Sie das Feld `ReconnectDelay` nur, wenn der Wert des Feldes `Reason` im Callback-Kontext `MQRC_RECONNECTING` ist.

Bei der Eingabe im Ereignishandler ist der Wert von `ReconnectDelay` die Anzahl der Millisekunden, die der Warteschlangenmanager wartet, bevor er versucht, die Verbindung wiederherzustellen.Tabelle 479 auf Seite 268 In sind die Werte aufgeführt, die Sie festlegen können, um das Verhalten des Warteschlangenmanagers bei der Rückgabe vom Ereignishandler zu ändern.

Tabelle 479. Werte für `ReconnectDelay`

Name	Wert	Beschreibung
`MQRD_NO_RECONNECT`	`-1`	Keine Verbindungswiederholung. Ein Fehler wird zur Anwendung zurückgegeben.

Tabelle 479. Werte für ReconnectDelay (Forts.)		
Name	**Wert**	**Beschreibung**
MQRD_NO_DELAY	0	Sofort versuchen, die Verbindung wiederherzustellen.
Milliseconds	>0	Es soll für diese Anzahl Millisekunden gewartet werden, bevor versucht wird, die Verbindung wiederherzustellen.

Anfangswerte und Sprachendeklarationen für MQCB

Callback-Kontextstruktur - Anfangswerte

Es gibt keine Anfangswerte für die **MQCBC**-Struktur. Die Struktur wird als Parameter an eine Callback-Routine übergeben. Der Warteschlangenmanager initialisiert die Struktur; sie wird nie von Anwendungen initialisiert.

Deklaration in Programmiersprache C

Callback-Kontextstruktur - Deklaration in der Programmiersprache C

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
   MQCHAR4   StrucId;                /* Structure identifier */
   MQLONG    Version;                /* Structure version number */
   MQLONG    CallType;               /* Why Function was called */
   MQHOBJ    Hobj;                   /* Object Handle */
   MQPTR     CallbackArea;           /* Callback data passed to the function */
   MQPTR     ConnectionArea;         /* MQCTL data area passed to the function */
   MQLONG    CompCode;               /* Completion Code */
   MQLONG    Reason;                 /* Reason Code */
   MQLONG    State;                  /* Consumer State */
   MQLONG    DataLength;             /* Message Data Length */
   MQLONG    BufferLength;           /* Buffer Length */
   MQLONG    Flags;                  /* Flags containing information about
                                        this consumer */
  /* Ver:1 */
   MQLONG    ReconnectDelay;         /* Number of milliseconds before */
  /* Ver:2 */  };                    /* reconnect attempt */
```

COBOL-Delaration

```
** 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 **
```

Deklaration in PL/I

```
dcl
 1 MQCBC based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version */
  3 CallType            fixed bin(31), /* Callback type */
  3 Hobj                fixed bin(31), /* Object Handle */
  3 CallbackArea        pointer,       /* User area passed to the function */
  3 ConnectionArea      pointer,       /* Connection User Area */
  3 CompCode            fixed bin(31); /* Completion Code */
  3 Reason              fixed bin(31); /* Reason Code */
  3 State               fixed bin(31); /* Consumer State */
  3 DataLength          fixed bin(31); /* Message Data Length */
  3 BufferLength        fixed bin(31); /* Message Buffer length */
  3 Flags               fixed bin(31); /* Consumer Flags */
/* Ver:1 */
  3 ReconnectDelay      fixed bin(31); /* Number of milliseconds before */
/* Ver:2 */                           /* reconnect attempt */
```

Deklaration in High Level Assembler

```
    MQCBC                 DSECT
    MQCBC                 DS  0F    Force fullword alignment
    MQCBC_STRUCID         DS  CL4   Structure identifier
    MQCBC_VERSION         DS  F     Structure version number
    MQCBC_CALLTYPE        DS  F     Why Function was called
    MQCBC_HOBJ            DS  F     Object Handle
    MQCBC_CALLBACKAREA    DS  A     Callback data passed to the function
    MQCBC_CONNECTIONAREA  DS  A     MQCTL Data area passed to the function
    MQCBC_COMPCODE        DS  F     Completion Code
    MQCBC_REASON          DS  F     Reason Code
    MQCBC_STATE           DS  F     Consumer State
    MQCBC_DATALENGTH      DS  F     Message Data Length
    MQCBC_BUFFERLENGTH    DS  F     Buffer Length
    MQCBC_FLAGS           DS  F     Flags containing information about this consumer
    MQCBC_RECONNECTDELAY  DS  F     Number of milliseconds before reconnect
    MQCBC_LENGTH          EQU *-MQCBC
                          ORG       MQCBC
    MQCBC_AREA            DS  CL(MQCBC_LENGTH)
```

MQCBD - Callback descriptor

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. Struktur, die die Callback-Funktion angibt.

Tabelle 480. Felder in MQCBD

Feld	Beschreibung	Thema
StrucID	Struktur-ID	StrucID
Version	Strukturversionsnummer	Version
CallbackType	Typ der Callback-Funktion	CallbackType
Options	Optionen zum Steuern der Nachrichtenverarbeitung	Options
Callback Area	Feld für zu verwendende Callback-Funktion	CallbackArea
CallbackFunction	Gibt an, ob die Funktion als API-Aufruf aufgerufen wird	CallbackFunction
CallbackName	Gibt an, ob die Funktion als dynamisch verknüpftes Programm aufgerufen wird	CallbackName
MaxMsgLength	Länge der längsten Nachricht, die gelesen werden kann	MaxMsgLength

Überblick für MQCBD

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSund WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQCBD-Struktur wird verwendet, um eine Callback-Funktion und die Optionen anzugeben, die ihre Verwendung durch den Warteschlangenmanager steuern.

Die Struktur ist ein Eingabeparameter im Aufruf MQCB.

Version: Die aktuelle Version von MQCBD ist MQCBD_VERSION_1.

Zeichensatz und Codierung: Daten in MQCBD müssen im Zeichensatz vorliegen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird, sowie in der Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQCBD

Alphabetische Liste der Felder für die MQCBD-Struktur.

Die MQCBD-Struktur enthält die folgenden Felder, die in alphabetischer Reihenfolge beschrieben werden:

CallbackArea (MQPTR)
Callback-Deskriptorstruktur - Feld CallbackArea

Dies ist ein Feld, das von einer Callback-Funktion verwendet werden kann.

Der Warteschlangenmanager trifft keine Entscheidungen auf der Basis des Inhalts dieses Felds und es wird unverändert aus dem Feld „CallbackArea (MQPTR)" auf Seite 263 in der MQCBC-Struktur übergeben, bei der es sich um einen Parameter in der Callback-Funktionsdeklaration handelt.

Der Wert wird nur bei einer *Operation* mit einem Wert MQOP_REGISTER ohne aktuell definierten Callback verwendet, er ersetzt keine frühere Definition.

Dies ist ein Ein-/Ausgabefeld für die Callback-Funktion. Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

CallbackFunction (MQPTR)
Callback-Deskriptorstruktur - Feld CallbackFunction

Die Callback-Funktion wird als Funktionsaufruf aufgerufen.

Über dieses Feld können Sie einen Zeiger zur Callback-Funktion angeben.

Sie müssen *entweder CallbackFunction* oder *CallbackName* angeben. Wenn Sie beides angeben, wird der Ursachencode MQRC_CALLBACK_ROUTINE_ERROR zurückgemeldet.

Ist weder *CallbackName* noch *CallbackFunction* gesetzt, schlägt der Aufruf mit dem Ursachencode MQRC_CALLBACK_ROUTINE_ERROR fehl.

Diese Option wird in der folgenden Umgebung nicht unterstützt: Programmiersprachen und Compiler, die keine Funktionszeigerverweise unterstützen. In diesen Situationen schlägt der Aufruf mit dem Ursachencode MQRC_CALLBACK_ROUTINE_ERROR fehl.

Unter z/OS wird die Funktion eventuell über Betriebssystem-Verbindungskonventionen aufgerufen. Geben Sie z. B. in der Programmiersprache C Folgendes an:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

Anmerkung: Bei der Verwendung von CICS mit WebSphere MQ V7.0.1 wird eine asynchrone Verarbeitung unterstützt, wenn:

- Apar PK66866 auf CICS TS 3.2 angewendet wird
- Apar PK89844 auf CICS TS 4.1 angewendet wird

CallbackName (MQCHAR128)
Callback-Deskriptorstruktur - Feld CallbackName

Die Callback-Funktion wird als dynamisch verknüpftes Programm aufgerufen.

Sie müssen *entweder* *CallbackFunction* oder *CallbackName* angeben. Wenn Sie beides angeben, wird der Ursachencode MQRC_CALLBACK_ROUTINE_ERROR zurückgemeldet.

Ist weder *CallbackName* noch *CallbackFunction* gesetzt, schlägt der Aufruf mit dem Ursachencode MQRC_CALLBACK_ROUTINE_ERROR fehl.

Das Modul wird geladen, wenn die erste zu verwendende Callback-Routine registriert wird, und entladen, wenn die Registrierung der letzten Callback-Routine, die es verwendet, aufgehoben wird.

Sofern im folgenden Text nicht anders angegeben, wird der Name innerhalb des Felds links ausgerichtet, wobei innerhalb des Namens keine Leerzeichen eingefügt werden. Der Name des Felds selbst wird mit Leerzeichen auf die Länge des Felds aufgefüllt. In den nachfolgenden Beschreibungen kennzeichnen eckige Klammern ([ ]) optionale Informationen:

IBM i
Der Callback-Name kann eines der folgenden Formate haben:

- Bibliothek "/" Programm
- Bibliothek "/" ServiceProgram "("FunctionName")"

Zum Beispiel `MyLibrary/MyProgram(MyFunction)`.

Der Bibliotheksname kann *LIBL sein. Der Bibliotheks- und der Programmname dürfen maximal 10 Zeichen lang sein.

UNIX-Systeme
Der Callback-Name ist der Name eines dynamisch ladbaren Moduls bzw. einer Bibliothek, dem als Suffix der Name einer Funktion angefügt wird, die sich in dieser Bibliothek befindet. Der Funktionsname muss in Klammern eingeschlossen sein. Dem Bibliotheksnamen kann optional ein Verzeichnispfad vorangestellt werden:

```
[path]library(function)
```

Ist der Pfad nicht angegeben, wird der Systemsuchpfad verwendet.

Der Name darf maximal 128 Zeichen lang sein.

Windows
Der Callback-Name ist der Name einer DLL (Dynamic-Link Library), dem als Suffix der Name einer Funktion angefügt wird, die sich in dieser Bibliothek befindet. Der Funktionsname muss in Klammern eingeschlossen sein. Dem Bibliotheksnamen kann optional ein Verzeichnispfad und ein Laufwerk vorangestellt werden:

```
[d:][path]library(function)
```

Sind das Laufwerk und der Pfad nicht angegeben, wird der Systemsuchpfad verwendet.

Der Name darf maximal 128 Zeichen lang sein.

z/OS
Der Callback-Name ist der Name eines Lademoduls, der für Spezifikationen im EP-Parameter des LINK- oder LOAD-Makros verwendet werden kann.

Der Name darf maximal 8 Zeichen lang sein.

z/OS CICS
Der Callback-Name ist der Name eines Lademoduls, der für Spezifikationen im PROGRAM-Parameter des Befehlsmakros EXEC CICS LINK verwendet werden kann.

Der Name darf maximal 8 Zeichen lang sein.

Das Programm kann über die Option REMOTESYTEM der installierten PROGRAM-Definition oder vom Programm für dynamisches Routing als "fern" definiert werden.

Die ferne CICS-Region muss mit WebSphere MQ verbunden sein, wenn vom Programm WebSphere MQ API-Aufrufe verwendet werden sollen. Beachten Sie jedoch, dass das Feld „Hobj (MQHOBJ)” auf Seite 267 in der MQCBC-Struktur in einem fernen System nicht gültig ist.

Wenn beim Laden von *CallbackName* ein Fehler auftritt, wird einer der nachstehenden Fehlercodes an die Anwendung zurückgegeben:

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

Außerdem wird eine Meldung in das Fehlerprotokoll geschrieben, das den Namen des Moduls enthält, für das der Ladevorgang versucht wurde, sowie der betreffende Ursachencode vom Betriebssystem.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist eine Nullzeichenfolge oder Leerzeichen.

CallbackType (MQLONG)
Callback-Deskriptorstruktur - Feld CallbackType

Dies ist der Typ der Callback-Funktion. Der Parameter muss einen der folgenden Werte haben:

MQCBT_MESSAGE_CONSUMER
Definiert diesen Callback als Nachrichtenkonsumentenfunktion.

Eine Callback-Funktion für einen Nachrichtenkonsumenten wird aufgerufen, wenn eine Nachricht, die die angegebenen Kriterien erfüllt, in einer Objektkennung vorhanden ist und die Verbindung gestartet wurde.

MQCBT_EVENT_HANDLER
Definiert diesen Callback als asynchrone Ereignisroutine; der Parameter dient nicht der Verarbeitung von Nachrichten für eine Kennung.

Hobj wird im MQCB-Aufruf zur Definition der Ereigniskennung nicht benötigt und wird ignoriert, wenn es angegeben ist.

Die Ereigniskennung wird bei Bedingungen aufgerufen, die sich auf die gesamte Umgebung des Nachrichtenkonsumenten auswirken. Die Konsumentenfunktion wird ohne Nachricht bei Eintreten eines Ereignisses aufgerufen, etwa wenn der Warteschlangenmanager oder die Verbindung beendet wird oder in den Quiescemodus wechselt. Sie wird nicht bei Bedingungen aufgerufen, die nur einen einzelnen Nachrichtenkonsumenten betreffen, z. B. MQRC_GET_INHIBITED.

Ereignisse werden an die Anwendung übergeben, unabhängig davon, ob die Verbindung gestartet oder gestoppt ist, mit Ausnahme der folgenden Umgebungen:

- CICS in der z/OS-Umgebung
- Anwendungen ohne Thread

Übergibt der Aufrufende keinen dieser Werte, schlägt der Aufruf mit dem *Reason*-Code MQRC_CALLBACK_TYPE_ERROR fehl.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCBT_MESSAGE_CONSUMER.

MaxMsgLength (MQLONG)
Dies ist die Länge der längsten Nachricht in Byte, die aus der Kennung ausgelesen und an die Callback-Routine übergeben werden kann. Callback-Deskriptorstruktur-Feld „MaxMsgLength“

Ist eine Nachricht länger, empfängt die Callback-Routine *MaxMsgLength*-Bytes der Nachricht sowie den Ursachencode:

- MQRC_TRUNCATED_MSG_FAILED oder
- MQRC_TRUNCATED_MSG_ACCEPTED, wenn Sie MQGMO_ACCEPT_TRUNCATED_MSG angegeben haben.

Die tatsächliche Nachrichtenlänge wird im Feld „DataLength (MQLONG)” auf Seite 266 der MQCBC-Struktur angegeben.

Der folgende spezielle Wert ist definiert:

MQCBD_FULL_MSG_LENGTH
Die Puffergröße wird vom System so angepasst, dass zurückgemeldete Nachrichten nicht abgeschnitten werden.

Wenn nicht genügend Speicher vorhanden ist, um einen Puffer für den Empfang der Nachricht anzulegen, ruft das System mit dem Ursachencode MQRC_STORAGE_NOT_AVAILABLE die Callback-Funktion auf.

Wenn Sie zum Beispiel eine Datenkonvertierung anfordern und unzureichender Speicherplatz zur Konvertierung der Nachrichtendaten vorhanden ist, wird die unkonvertierte Nachricht an die Callback-Funktion übergeben.

Dies ist ein Eingabefeld. Der Anfangswert des Felds *MaxMsgLength* ist MQCBD_FULL_MSG_LENGTH.

Optionen (MQLONG)
Callback-Deskriptorstruktur - Feld Options

Eine oder alle der nachstehenden Optionen können angegeben werden. Wenn mehr als eine Option erforderlich ist, können die Werte:

- gemeinsam hinzugefügt werden (dieselbe Konstante nicht mehr als einmal hinzufügen) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

MQCBDO_FAIL_IF_QUIESCING
Der MQCB-Aufruf schlägt fehl, wenn der Warteschlangenmanager sich im Quiescestatus befindet.

Unter z/OS erzwingt diese Option auch dann ein Fehlschlagen des MQCB-Aufrufs, wenn sich die Verbindung (für eine CICS- oder IMS-Anwendung) im Quiescestatus befindet.

Geben Sie MQGMO_FAIL_IF_QUIESCING in den an den MQCB-Aufruf übergebenen MQGMO-Optionen an, um einen Hinweis an die Nachrichtenkonsumenten auszulösen, wenn sie in den Quiescemodus versetzt werden.

Steuerungsoptionen: Die folgenden Optionen steuern, ob die Callback-Funktion aufgerufen wird, ohne eine Nachricht, wenn sich der Status des Konsumenten ändert:

MQCBDO_REGISTER_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_REGISTER_CALL aufgerufen.

MQCBDO_START_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_START_CALL aufgerufen.

MQCBDO_STOP_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_STOP_CALL aufgerufen.

MQCBDO_DEREGISTER_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_DEREGISTER_CALL aufgerufen.

MQCBDO_EVENT_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_EVENT_CALL aufgerufen.

MQCBDO_MC_EVENT_CALL
Die Callback-Funktion wird mit dem Aufruftyp MQCBCT_MC_EVENT_CALL aufgerufen.

Weitere Informationen über diese Aufruftypen finden Sie im Abschnitt „CallType (MQLONG)” auf Seite 264.

Standardoption: Falls Sie keine der beschriebenen Optionen benötigen, verwenden Sie folgende Option:

MQCBDO_NONE
Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an.

MQCBDO_NONE dient zur Unterstützung der Programmdokumentation und sollte nicht zusammen mit anderen Optionen verwendet werden. Da sie jedoch den Wert null hat, kann ihre Verwendung nicht erkannt werden.

Dies ist ein Eingabefeld. Der Anfangswert des Felds *Options* ist MQCBDO_NONE.

StrucId (MQCHAR4)
Callback-Deskriptorstruktur - StrucId-Feld

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQCBD_STRUC_ID
ID für die Struktur des Callback-Deskriptors.

Für die Programmiersprache C ist auch die Konstante MQCBD_STRUC_ID_ARRAY definiert; diese Konstante hat denselben Wert wie MQCBD_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCBD_STRUC_ID.

Version (MQLONG)
Callback-Deskriptorstruktur - Feld Version

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQCBD_VERSION_1
Callback-Deskriptorstruktur Version-1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCBD_CURRENT_VERSION
Aktuelle Version der Callback-Deskriptorstruktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCBD_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQCBD
Callback-Deskriptorstruktur - Anfangswerte

Tabelle 481. Anfangswerte der Felder in MQCBD

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCBD_STRUC_ID	'CBD¬'
Version	MQCBD_VERSION_1	1
CallBackType	MQCBT_MESSAGE_CONSUMER	1
Options	MQCBDO_NONE	0
CallbackArea	--	Nullzeiger oder null Leerzeichen
CallbackFunction	--	Nullzeiger oder null Leerzeichen
CallbackName	--	Nullzeichenfolge oder Leerzeichen.
MaxMsgLength	MQCBD_FULL_MSG_LENGTH	-1

Tabelle 481. Anfangswerte der Felder in MQCBD (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert Nullzeichenfolge oder Leerzeichen stellt die Nullzeichenfolge in der Programmiersprache C und Leerzeichen in anderen Programmiersprachen dar.
3. In der Programmiersprache C enthält die MakrovariableMQCBD_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQCBD MyCBD = {MQCBD_DEFAULT};
```

Deklaration in Programmiersprache C

Callback-Deskriptorstruktur - Deklaration für Programmiersprache C

```
typedef struct tagMQCBD MQCBD;
 struct tagMQCBD {
   MQCHAR4    StrucId;                /* Structure identifier */
   MQLONG     Version;                /* Structure version number */
   MQLONG     CallBackType;           /* Callback function type */
   MQLONG     Options;                /* Options controlling message
                                         consumption */
   MQPTR      CallbackArea;           /* User data passed to the function */
   MQPTR      CallbackFunction;       /* Callback function pointer */
   MQCHAR128  CallbackName;           /* Callback name */
   MQLONG     MaxMsgLength;           /* Maximum message length */
 };
```

COBOL-Delaration

```
** 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.
```

Deklaration in PL/I

```
dcl
 1 MQCBD based,
  3 StrucId             char(4),       /* Structure identifier*/
  3 Version             fixed bin(31), /* Structure version*/
  3 CallbackType        fixed bin(31), /* Callback function type */
  3 Options             fixed bin(31), /* Options */
  3 CallbackArea        pointer,       /* User area passed to the function */
  3 CallbackFunction    pointer,       /* Callback Function Pointer */
  3 CallbackName        char(128),     /* Callback Program Name */
  3 MaxMsgLength        fixed bin(31); /* Maximum Message Length */
```

MQCHARV - Zeichenfolge variabler Länge

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur.

Feld	Beschreibung	Thema
VSPtr	Verweis auf die Zeichenfolge variabler Länge	VSPtr
VSOffset	Abstand der Zeichenfolge variabler Länge vom Start der Struktur, die diese MQCHARV-Struktur enthält, in Bytes	VSOffset
VSLength	Die Länge der durch das Feld VSPtr oder VSOffset adressierten Zeichenfolge variabler Länge in Bytes.	VSLength
VSBufSize	Die Größe des durch das Feld VSPtr oder VSOffset adressierten Puffers in Bytes.	VSBufSize
VSCCSID	Die Zeichensatzkennung der durch das Feld VSPtr oder VSOffset adressierten Zeichenfolge variabler Länge.	VSCCSID

Überblick für MQCHARV

Verfügbarkeit: AIX, HP-UX, Solaris, Linux, IBM i, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Sie verwenden die MQCHARV-Struktur, um eine Zeichenfolge variabler Länge zu beschreiben.

Zeichensatz und Codierung: Daten in der MQCHARV-Struktur müssen in der Codierung des lokalen Warteschlangenmanagers vorliegen, der durch MQENC_NATIVE angegeben wird, sowie im Zeichensatz des Felds VSCCSID innerhalb der Struktur. Wenn die Anwendung als MQ-Client ausgeführt wird, muss die Struktur die Codierung des Clients aufweisen. Einige Zeichensätze haben eine Darstellung, die von der Codierung abhängig ist. Wenn VSCCSID einer dieser Zeichensätze ist, ist die verwendete Codierung mit der Codierung der anderen Felder in der MQCHARV-Struktur identisch. Der durch VSCCSID identifizierte Zeichensatz kann ein Doppelbytezeichensatz sein.

Verwendung: Die MQCHARV-Struktur adressiert Daten, die mit der Struktur, in der sie enthalten sind, möglicherweise nicht verknüpft sind. Um diese Daten zu adressieren, können Felder verwendet werden, die mit dem Zeigerdatentyp deklariert wurden. Beachten Sie, dass COBOL den Zeigerdatentyp nicht in allen Umgebungen unterstützt. Daher können die Daten auch mit Feldern adressiert werden, die den Abstand der Daten vom Start der Struktur enthalten, die die MQCHARV-Struktur enthält.

Programmierung in der Programmiersprache COBOL

Wenn Sie eine Anwendung zwischen Umgebungen portieren möchten, müssen Sie prüfen, ob der Zeigerdatentyp in allen beabsichtigten Umgebungen verfügbar ist. Andernfalls muss die Anwendung die Daten mit den Abstandsfeldern anstatt mit den Zeigerfeldern adressieren.

In Umgebungen, in denen Zeiger nicht unterstützt werden, können Sie die Zeigerfelder als Bytezeichenfolgen mit der entsprechenden Länge deklarieren. Dabei ist der Anfangswert die Bytezeichenfolge, die ausschließlich aus Nullen besteht. Ändern Sie diesen Anfangswert nicht, wenn Sie die Abstandsfelder verwenden. Eine Möglichkeit, dies zu tun, ohne die mitgelieferten Copybooks zu ändern, besteht in der Verwendung von:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

wobei CMQCHRVV gegen das zu verwendende Copybook ausgetauscht werden kann.

Felder für MQCHARV

Die MQCHARV-Struktur enthält die folgenden Felder; die Felder werden in **Alphabetische Reihenfolge** beschrieben:

VSBufSize (MQLONG)
Dies ist die Größe (in Bytes) des Puffers, der im Feld "VSPtr" oder "VSOffset" angegeben wird.

Wenn die MQCHARV-Struktur als Ausgabefeld in einem Funktionsaufruf verwendet wird, dann muss dieses Feld mit der Länge des Puffers initialisiert werden, die angegeben wurde. Wenn der Wert von VSLength größer als der Wert von VSBufSize ist, dann wird im Puffer nur die in VSBufSize angegebene Menge an Datenbytes an den Aufrufer zurückgegeben.

Dieser Wert muss größer-gleich null sein oder es muss der folgende Sonderwert angegeben werden, der erkannt wird:

MQVS_USE_VSLENGTH
Bei Angabe dieses Werts wird die Länge des Puffers dem Feld "VSLength" in der MQCHARV-Struktur entnommen. Verwenden Sie diesen Wert nicht, wenn die Struktur als Ausgabefeld verwendet wird und ein Puffer angegeben ist.

Dies ist der Anfangswert dieses Felds.

VSCCSID (MQLONG)
Dies ist die Zeichensatzkennung der durch das Feld VSPtr oder VSOffset adressierten Zeichenfolge variabler Länge.

Der Anfangswert dieses Feldes ist MQCCSI_APPL. Dieser Wert wird von MQ definiert, um anzugeben, dass er in die tatsächliche Zeichensatzkennung des aktuellen Prozesses geändert werden sollte. Infolgedessen wird der MQCCSI_APPL nie einer Zeichenfolge variabler Länge zugeordnet. Der Anfangswert dieses Feldes kann geändert werden, indem ein anderer Wert für die Konstante MQCCSI_APPL für Ihre Kompilierungseinheit mit den entsprechenden Verfahren für die Programmiersprache Ihrer Anwendung definiert wird.

VSLength (MQLONG)
Die Länge der durch das Feld VSPtr oder VSOffset adressierten Zeichenfolge variabler Länge in Bytes.

Der Anfangswert dieses Feldes ist 0. Der Wert muss größer-gleich null oder der folgende Sonderwert sein, der erkannt wird:

MQVS_NULL_TERMINATED

Wenn MQVS_NULL_TERMINATED nicht angegeben ist, sind VSLength-Bytes als Bestandteil der Zeichenfolge enthalten. Wenn Nullzeichen vorhanden sind, begrenzen diese die Zeichenfolge nicht.

Wenn MQVS_NULL_TERMINATED angegeben ist, wird die Zeichenfolge durch die erste in der Zeichenfolge vorkommende Null begrenzt. Die Null selbst ist nicht als Bestandteil dieser Zeichenfolge enthalten.

Anmerkung: Das Nullzeichen, das bei Angabe von MQVS_NULL_TERMINATED zum Beenden einer Zeichenfolge verwendet wird, ist eine Null aus dem durch VSCCSID angegebenen codierten Zeichensatz.

In UTF-16 (UCS-2-CCSIDs 1200 und 13488) ist dies beispielsweise die aus zwei Bytes bestehende Unicode-Codierung, bei der eine Null durch eine 16-Bit-Zahl aller Nullen dargestellt wird. In UTF-16 sind Einzelbytes häufig auf Nullen festgelegt, die Bestandteil von Zeichen sind (z. B. 7-Bit-ASCII-Zeichen), die Zeichenfolgen werden jedoch nur mit einer Null beendet, wenn zwei Nullbytes an einer geraden Bytegrenze vorhanden sind. An einer ungeraden Grenze können sich zwei "Null"-Byte befinden, wenn beide Teil von gültigen Zeichen sind. Beispiel: x'01' x'00 x'00' x'30' stellt zwei gültige Unicode-Zeichen dar und beendet die Zeichenfolge nicht mit einer Null.

VSOffset (MQLONG)
Der Offset kann positiv oder negativ sein. Sie können das Feld VSPtr oder das Feld VSOffset verwenden, um die Zeichenfolge variabler Länge anzugeben, jedoch nicht beide Felder. Der Offset in Bytes der Zeichenfolge variabler Länge ab dem Beginn des MQCHARV oder der Struktur, die sie enthält.

Ist die MQCHARV-Struktur in eine andere Struktur eingebettet, ist dieser Wert der Offset (in Byte) der Zeichenfolge mit variabler Länge vom Anfang der Struktur, die diese MQCHARV-Struktur enthält. Ist

die MQCHARV-Struktur nicht in eine andere Struktur eingebettet, etwa wenn sie als Parameter eines Funktionsaufrufs angegeben ist, bezieht sich der Offset auf den Anfang der MQCHARV-Struktur.

Der Anfangswert dieses Feldes ist 0.

VSPtr (MQPTR)
Dies ist ein Zeiger auf die Zeichenfolge variabler Länge.

Sie können das Feld VSPtr oder das Feld VSOffset verwenden, um die Zeichenfolge variabler Länge anzugeben, jedoch nicht beide Felder.

Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

Anfangswerte und Sprachendeklarationen für MQCHARV

Anfangswerte der Felder in MQCHARV

Name des Felds	Name der Konstante	Wert der Konstanten
VSPtr	--	Nullzeiger oder null Byte.
VSOffset	--	0
VSBufSize	MQVS_USE_VSLENGTH	0
VSLength	--	0
VSCCSID	MQCCSI_APPL	-3

Anmerkung: In der Programmiersprache C enthält die Makrovariable MQCHARV_DEFAULT die oben aufgeführten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQCHARV MyVarStr = {MQCHARV_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQCHARV MQCHARV;
struct tagMQCHARV {
  MQPTR     VSPtr;              /* Address of variable length string */
  MQLONG    VSOffset;           /* Offset of variable length string */
  MQLONG    VSBufSize;          /* Size of buffer */
  MQLONG    VSLength;           /* Length of variable length string */
  MQLONG    VSCCSID;            /* CCSID of variable length string */
};
```

COBOL-Deklaration für 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.
```

Deklaration in PL/I

```
dcl
 1 MQCHARV based,
  3 VSPtr       pointer,       /* Address of variable length string */
  3 VSOffset    fixed bin(31), /* Offset of variable length string */
  3 VSBufSize   fixed bin(31), /* Size of buffer */
  3 VSLength    fixed bin(31), /* Length of variable length string */
  3 VSCCSID     fixed bin(31); /* CCSID of variable length string */
```

Deklaration in High Level Assembler

```
MQCHARV             DSECT
MQCHARV_VSPTR       DS   F     Address of variable length string
MQCHARV_VSOFFSET    DS   F     Offset of variable length string
MQCHARV_VSBUFSIZE   DS   F     Size of buffer
MQCHARV_VSLENGTH    DS   F     Length of variable length string
MQCHARV_VSCCSID     DS   F     CCSID of variable length string
*
MQCHARV_LENGTH      EQU  *-MQCHARV
                    ORG  MQCHARV
MQCHARV_AREA        DS   CL(MQCHARV_LENGTH)
```

Neudefinition von MQCCSI_APPL

Die folgenden Beispiele zeigen, wie Sie den Wert von MQCCSI_APPL in verschiedenen Programmiersprachen überschreiben können. Sie können den Wert von MQCCSI_APPL ändern, indem Sie die Notwendigkeit entfernen, die VSCCSID für jede Zeichenfolge variabler Länge separat festzulegen.

In diesen Beispielen ist die CCSID auf 1208 festgelegt. Ändern Sie diesen Wert in den Wert, den Sie benötigen. Dieser Wert wird der Standardwert, den Sie durch Festlegen der VSCCSID in einer bestimmten Instanz von MQCHARV überschreiben können.

C-Syntax

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

COBOL-Syntax

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

PL/I-Syntax

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

System/390-Assemblersyntax

```
MQCCSI_APPL EQU 1208
         CMQA LIST=NO
```

MQCIH - Header für CICS-Bridge

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 482. Felder in MQCIH

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version

Tabelle 482. Felder in MQCIH (Forts.)		
Feld	**Beschreibung**	**Thema**
StrucLength	Länge der MQCIH-Struktur	StrucLength
Encoding	Reserved	Encoding
CodedCharSetId	Reserved	CodedCharSetId
Format	MQ-Formatname der Daten, die auf MQCIH folgen	Format
Flags	Markierungen	Flags
ReturnCode	Rückkehrcode von der Bridge	ReturnCode
CompCode	MQ-Beendigungscode oder CICS EIBRESP	CompCode
Reason	MQ-Ursachen- bzw. MQ-Rückmeldungscode oder CICS EIBRESP2	Reason
UOWControl	Steuerung der Arbeitseinheit	UOWControl
GetWaitInterval	Warteintervall für den von der Bridge-Task ausgegebenen MQGET-Aufruf	GetWaitInterval
LinkType	Verbindungstyp	LinkType
OutputDataLength	Ausgegebene Datenlänge des Kommunikationsbereichs	OutputDataLength
FacilityKeepTime	Freigabezeit der Bridge-Funktion	FacilityKeepTime
ADSDescriptor	ADS-Deskriptor zum Senden/Empfangen	ADSDescriptor
ConversationalTask	Gibt an, ob die Task interaktiv sein kann	ConversationalTask
TaskEndStatus	Status am Ende der Task	TaskEndStatus
Facility	Bridge-Funktionstoken	Facility
Function	MQ -Aufrufname oder CICS -Funktion EIBFN	Function
AbendCode	Abbruchcode	AbendCode
Authenticator	Kennwort oder Passticket	Authenticator
Reserved1	Reserved	Reserved1
ReplyToFormat	Name des MQ-Formats der Antwortnachricht	ReplyToFormat
RemoteSysId	Zu verwendende ID des fernen CICS-Systems	RemoteSysId
RemoteTransId	Zu verwendende CICS-RTRANSID	RemoteTransId
TransactionId	Anzuhängende Transaktion	TransactionId
FacilityLike	Vom Terminal emulierte Attribute	FacilityLike
AttentionId	AID-Taste	AttentionId
StartCode	Startcode der Transaktion	StartCode
CancelCode	Abbruchtransaktionscode	CancelCode
NextTransactionId	Nächste anzuhängende Transaktion	NextTransactionId
Reserved2	Reserved	Reserved2
Reserved3	Reserved	Reserved3

Tabelle 482. Felder in MQCIH (Forts.)		
Feld	**Beschreibung**	**Thema**
Anmerkung: Die übrigen Felder sind nicht vorhanden, wenn *Version* kleiner als MQCIH_VERSION_2 ist.		
CursorPosition	Cursorposition	CursorPosition
ErrorOffset	Relative Position des Fehlers in der Nachricht	ErrorOffset
InputItem	Reserved	InputItem
Reserved4	Reserved	Reserved4

Übersicht über MQCIH

Verfügbarkeit: AIX, HP-UX, z/OS, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQCIH-Struktur beschreibt die Informationen, die am Anfang einer Nachricht, die über WebSphere MQ for z/OSan die CICS -Brücke gesendet wird, vorhanden sein können.

Formatname: MQFMT_CICS.

Version: Die aktuelle Version von MQCIH lautet MQCIH_VERSION_2. Felder, die nur in der neueren Version der Struktur verfügbar sind, sind in den folgenden Beschreibungen gekennzeichnet.

Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützten Programmiersprachen bereitgestellt werden, enthalten die aktuellste Version von MQCIH, wobei der Anfangswert des Felds *Version* auf MQCIH_VERSION_2 gesetzt ist.

Zeichensatz und Codierung: Für den Zeichensatz und die Codierung, die für die MQCIH-Struktur und Anwendungsnachrichtendaten verwendet werden, gelten besondere Bedingungen:

- Anwendungen, die Verbindungen mit dem Warteschlangenmanager herstellen, der Eigner der CICS-Brücken-WS ist, müssen eine MQCIH-Struktur mit Zeichensatz und Codierung des Warteschlangenmanagers bereitstellen. Der Grund hierfür ist, dass die Datenkonvertierung der MQCIH-Struktur in diesem Fall nicht ausgeführt wird.
- Anwendungen, die sich mit anderen Warteschlangenmanagern verbinden, können eine MQCIH-Struktur bereitstellen, die beliebige unterstützte Zeichensätze und Codierungen aufweisen kann; der empfangende Nachrichtenkanalagent, der mit dem als Eigner der CICS-Brückenwarteschlange fungierenden Warteschlangenmanager verbunden ist, konvertiert die MQCIH-Struktur.
- Die Anwendungsnachrichtendaten, die auf die MQCIH-Struktur folgen, müssen denselben Zeichensatz und dieselbe Codierung wie die MQCIH-Struktur aufweisen. Sie können den Zeichensatz und die Codierung der Anwendungsnachrichtendaten nicht mit den Feldern *CodedCharSetId* und *Encoding* in der MQCIH-Struktur angeben.

  Sie müssen einen Datenkonvertierungsexit bereitstellen, um die Anwendungsnachrichtendaten zu konvertieren, wenn die Daten nicht einem der integrierten Formate entsprechen, die vom Warteschlangenmanager unterstützt werden.

Syntax: Wenn die Anwendung Werte erfordert, die mit den in Tabelle 484 auf Seite 292 aufgeführten Anfangswerten identisch sind, und wenn die Bridge mit AUTH=LOCAL oder AUTH=IDENTIFY ausgeführt wird, können Sie die MQCIH-Struktur aus der Nachricht ausschließen. In allen anderen Fällen muss die Struktur vorhanden sein.

Die Bridge akzeptiert entweder eine MQCIH-Struktur der Version 1 oder 2. Für 3270-Transaktionen muss allerdings eine Struktur der Version 2 verwendet werden.

Die Anwendung muss sicherstellen, dass Felder, die als Anforderungsfelder dokumentiert sind, geeignete Werte in der an die Bridge gesendeten Nachricht enthalten; diese Felder dienen als Eingabe für die Bridge.

Felder, die als Antwortfelder dokumentiert sind, werden durch die CICS-Brücke in der Antwortnachricht festgelegt, die von der Brücke an die Anwendung gesendet wird. Fehlerinformationen werden in den Feldern *ReturnCode*, *Function*, *CompCode*, *Reason*und *AbendCode* zurückgegeben, aber nicht alle sind in allen Fällen festgelegt. In der Tabelle 483 auf Seite 283 ist angegeben, welche Felder für die verschiedenen Werte von *ReturnCode* festgelegt werden.

Tabelle 483. Inhalt der Felder mit Fehlerinformationen in der MQCIH-Struktur bei MQCIH

ReturnCode	***Function***	***CompCode***	***Reason***	***AbendCode***
MQCRC_OK	-	-	-	-
MQCRC_BRIDGE_ERROR	-	-	MQFB_CICS_*	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	Name des MQ-Aufrufs	MQ *CompCode*	MQ *Reason*	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	CICS ABCODE

Felder für MQCIH

Die MQCIH-Struktur enthält die folgenden Felder; die Felder werden in **Alphabetische Reihenfolge** beschrieben:

AbendCode (MQCHAR4)
AbendCode ist ein Antwortfeld. Die Länge dieses Felds wird durch MQ_ABEND_CODE_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

Der in diesem Feld zurückgegebene Wert ist nur von Bedeutung, wenn das Feld *ReturnCode* den Wert MQCRC_APPLICATION_ABEND oder MQCRC_BRIDGE_ABEND enthält. Ist dies der Fall, enthält *AbendCode* den CICS-Wert ABCODE.

ADSDescriptor (MQLONG)
Dieses Feld ist ein Indikator, der angibt, ob ADS-Deskriptoren für BMS-Anforderungen von SEND und RECEIVE gesendet werden sollen.

Die folgenden Werte sind definiert:

MQCADSD_NONE
Es werden ADS-Deskriptoren gesendet oder empfangen.

MQCADSD_SEND
Es werden ADS-Deskriptoren gesendet.

MQCADSD_RECV
Es werden ADS-Deskriptoren empfangen.

MQCADSD_MSGFORMAT
Für die ADS-Deskriptoren wird das Nachrichtenformat verwendet.

In diesem Fall werden die ADS-Deskriptoren mit der Langform des ADS-Deskriptors gesendet oder empfangen. Die Langform enthält Felder, die auf 4-Byte-Grenzen ausgerichtet sind.

Legen Sie das Feld *ADSDescriptor* wie folgt fest:

- Wenn Sie keine ADS-Deskriptoren verwenden, setzen Sie das Feld auf MQCADSD_NONE.
- Wenn Sie in jeder Umgebung ADS-Deskriptoren mit *derselben* CCSID verwenden, setzen Sie das Feld auf die Summe von MQCADSD_SEND und MQCADSD_RECV.

- Wenn Sie in jeder Umgebung ADS-Deskriptoren mit *unterschiedlichen* CCSIDs verwenden, setzen Sie das Feld auf die Summe von MQCADSD_SEND, MQCADSD_RECV, und MQCADSD_MSGFORMAT.

Dies ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Der Anfangswert dieses Felds lautet MQCADSD_NONE.

AttentionId (MQCHAR4)
Der Wert in diesem Feld bestimmt den Anfangswert der AID-Taste beim Start der Transaktion. Es handelt sich um einen 1-Byte-Wert, linksbündig.

AttentionId ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Die Länge dieses Felds wird durch MQ_ATTENTION_ID_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

Authenticator (MQCHAR8)
Der Wert dieses Feldes ist das Kennwort oder das Passticket.

Wenn für die CICS-Brücke eine Authentifizierung auf Basis der Benutzer-ID aktiv ist, wird *Authenticator* in Verbindung mit der Benutzer-ID im MQMD-Identitätskontext zur Authentifizierung des Nachrichtenabsenders verwendet.

Dies ist ein Anforderungsfeld. Die Länge des Felds wird durch MQ_AUTHENTICATOR_LENGTH angegeben. Der Anfangswert dieses Feldes ist 8 Leerstellen.

CancelCode (MQCHAR4)
Der Wert in diesem Feld ist der Abbruchcode, der für die Beendigung der Transaktion verwendet werden soll (normalerweise eine Dialogtransaktion, die weitere Daten anfordert). Andernfalls wird dieses Feld auf Leerzeichen gesetzt.

Dieses Feld ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Die Länge dieses Felds wird durch MQ_CANCEL_CODE_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

CodedCharSetId (MQLONG)
CodedCharSetId ist ein reserviertes Feld, dessen Wert nicht von Bedeutung ist. Der Anfangswert dieses Feldes ist 0.

Die Zeichensatz-ID für unterstützte Strukturen, die auf eine MQCIH-Struktur folgen, ist mit der Zeichensatz-ID der MQCIH-Struktur selbst identisch und wird aus einem beliebigen vorhergehenden WebSphere MQ-Header übernommen.

CompCode (MQLONG)
Dieses Feld ist ein Antwortfeld. Sein Anfangswert lautet MQCC_OK.

Der in diesem Feld zurückgegebene Wert hängt von *ReturnCode* ab; siehe Tabelle 483 auf Seite 283.

ConversationalTask (MQLONG)
Dieses Feld ist ein Indikator, der angibt, ob die Task Anforderungen für weitere Informationen ausgeben oder die Task stoppen und eine Abbruchnachricht ausgeben kann.

Als Wert muss eine der folgenden Optionen angegeben werden:

MQCCT_YES
Die Task ist interaktiv.

MQCCT_NO
Die Task ist nicht interaktiv.

Dieses Feld ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Der Anfangswert dieses Felds lautet MQCCT_NO.

CursorPosition (MQLONG)
Der Wert in diesem Feld enthält die Ausgangscursorposition beim Start der Transaktion. Bei Dialogtransaktionen befindet sich die Cursorposition im RECEIVE-Vektor.

Dieses Feld ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Der Anfangswert dieses Feldes ist 0. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCIH_VERSION_2 ist.

Encoding (MQLONG)
Bei diesem Feld handelt es sich um ein reserviertes Feld, dessen Wert nicht von Bedeutung ist. Der Anfangswert ist 0.

Die Codierung für unterstützte Strukturen, die auf eine MQCIH-Struktur folgen, ist mit der Codierung der MQCIH-Struktur selbst identisch und wird aus einem beliebigen vorhergehenden WebSphere MQ-Header übernommen.

ErrorOffset (MQLONG)
Das Feld "ErrorOffset" enthält die Position ungültiger Daten, die vom Bridge-Exit gefunden werden. Dieses Feld gibt den Offset vom Anfang der Nachricht an die Position der ungültigen Daten an.

ErrorOffset ist ein Antwortfeld, das nur für 3270-Transaktionen verwendet wird. Der Anfangswert dieses Feldes ist 0. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCIH_VERSION_2 ist.

Facility (MQBYTE8)
In diesem Feld wird das 8-Byte-Brückenfunktion-Token angezeigt.

Ein Bridge-Funktionstoken ermöglicht mehreren Transaktionen in einem Pseudodialog die Verwendung derselben Bridge-Funktion (virtuelles 3270-Terminal). Legen Sie in der ersten bzw. einzigen Nachricht in einem Pseudodialog den Wert MQCFAC_NONE fest. Dieser Wert weist CICS an, für diese Nachricht eine neue Brückenfunktion zuzuordnen. Ein Brückenfunktionstoken wird in Antwortnachrichten zurückgegeben, wenn in der Eingabenachricht für *FacilityKeepTime* ein Wert ungleich null angegeben ist. Nachfolgende Eingabenachrichten innerhalb eines Pseudodialogs müssen dann dasselbe Bridge-Funktionstoken verwenden.

Der folgende spezielle Wert ist definiert:

MQCFAC_NONE
Kein Funktionstoken angegeben.

Für die Programmiersprache C ist auch die Konstante MQCFAC_NONE_ARRAY definiert, die denselben Wert wie die Konstante MQCFAC_NONE hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Dieses Feld ist sowohl ein Anforderungs- als auch ein Antwortfeld, das nur für 3270-Transaktionen verwendet wird. Die Länge dieses Felds wird durch MQ_FACILITY_LENGTH angegeben. Der Anfangswert dieses Felds lautet MQCFAC_NONE.

FacilityKeepTime (MQLONG)
FacilityKeepTime gibt die Beibehaltungszeit der Brückenfunktion nach Beendigung der Benutzertransaktion in Sekunden an.

Geben Sie für pseudodialogfähige Transaktionen einen Wert an, der der erwarteten Dauer eines Pseudodialogs entspricht; geben Sie für die letzte Transaktion eines Pseudodialogs null an, und für andere Transaktionstypen ebenfalls null.

Dieses Feld ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Der Anfangswert dieses Feldes ist 0.

FacilityLike (MQCHAR4)
FacilityLike ist der Name eines installierten Terminals, das als Modell für die Brückenfunktion verwendet werden soll.

Werden Leerzeichen als Wert angegeben, wird *FacilityLike* der Definition des Bridge-Transaktionsprofils entnommen oder es wird ein Standardwert verwendet.

Dieses Feld ist ein Anforderungsfeld, das nur für 3270-Transaktionen verwendet wird. Die Länge dieses Felds wird durch MQ_FACILITY_LIKE_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

Flags (MQLONG)
Dieses Feld ist ein Anforderungsfeld. Der Anfangswert dieses Felds lautet MQCIH_NONE.

Folgende Werte sind möglich:

MQCIH_NONE
Keine Flags.

MQCIH_PASS_EXPIRATION
Die Antwortnachricht enthält:

- Die Verfallsberichtsoptionen, die auch in der Anforderungsnachricht zu finden sind.
- Die verbleibende Ablaufzeit aus der Anforderungsnachricht ohne Anpassung an die Verarbeitungszeit der Bridge.

Wenn Sie diesen Wert übergehen, wird eine *unbegrenzte* Ablaufzeit festgelegt.

MQCIH_REPLY_WITHOUT_NULLS
Die Länge der Antwortnachricht einer CICS-DPL-Programmanforderung wird dahingehend angepasst, dass keine abschließenden Nullen (X'00') am Ende des vom DPL-Programm zurückgegebenen Kommunikationsbereichs enthalten sind. Wird dieser Wert nicht festgelegt, sind die Nullen möglicherweise von Bedeutung und es wird der vollständige Kommunikationsbereich zurückgegeben.

MQCIH_SYNC_ON_RETURN
Bei der CICS-Verbindung für DPL-Anforderungen wird die Option SYNCONRETURN verwendet. Dadurch muss CICS bei Programmabschluss einen Synchronisationspunkt einbeziehen, wenn die Übermittlung an eine andere CICS-Region erfolgt. Die Brücke gibt nicht an, an welche CICS-Region die Anforderung übermittelt werden soll. Dies wird durch die CICS-Programmdefinition oder die Funktionen des Lastausgleichs gesteuert.

Format (MQCHAR8)
Dieses Feld enthält den WebSphere MQ-Formatnamen der Daten, die auf die MQCIH-Struktur folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds sind mit den Regeln identisch, die für die Codierung des Felds *Format* im MQMD gelten.

Dieser Formatname wird auch für die Antwortnachricht verwendet, wenn das Feld *ReplyToFormat* den Wert MQFMT_NONE aufweist.

- Bei DPL-Anforderungen muss *Format* dem Formatnamen des Kommunikationsbereichs entsprechen.
- Bei 3270-Anforderungen muss *Format* den Wert CSQCBDCI haben. Die Bridge setzt das Format für Antwortnachrichten auf CSQCBDC0.

Die Datenkonvertierungsexits für diese Formate müssen auf dem Warteschlangenmanager installiert sein, auf dem sie ausgeführt werden.

Wenn die Anforderungsnachricht eine Fehlerantwortnachricht generiert, hat die Fehlerantwortnachricht den Formatnamen MQFMT_STRING.

Dieses Feld ist ein Anforderungsfeld. Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

Function (MQCHAR4)
Dieses Feld ist ein Antwortfeld. Die Länge dieses Felds wird durch MQ_FUNCTION_LENGTH angegeben. Der Anfangswert dieses Felds lautet MQCFUNC_NONE.

Der in diesem Feld zurückgegebene Wert hängt von *ReturnCode* ab; siehe Tabelle 483 auf Seite 283. Die folgenden Werte sind möglich, wenn *Function* den Namen eines WebSphere MQ-Aufrufs enthält:

MQCFUNC_MQCONN
MQCONN-Aufruf.

MQCFUNC_MQGET
MQGET-Aufruf.

MQCFUNC_MQINQ
MQINQ-Aufruf.

MQCFUNC_MQOPEN
MQOPEN-Aufruf.

MQCFUNC_MQPUT
MQPUT-Aufruf.

MQCFUNC_MQPUT1
MQPUT1-Aufruf.

MQCFUNC_NONE
Kein Aufruf.

In allen Fällen sind für die Programmiersprache C die Konstanten MQCFUNC_*_ARRAY ebenfalls definiert; diese Konstanten enthalten dieselben Werte wie die entsprechenden Konstanten des Typs MQCFUNC_*, allerdings handelt es sich nicht um Zeichenfolgen, sondern um Feldgruppen von Zeichen.

GetWaitInterval (MQLONG)
Dieses Feld ist ein Anforderungsfeld. Sein Anfangswert lautet MQCGWI_DEFAULT.

Dieses Feld wird nur benötigt, wenn *UOWControl* den Wert MQCUOWC_FIRST aufweist. Es ermöglicht der sendenden Anwendung die Angabe der ungefähren Zeit in Millisekunden, die die von der Bridge ausgegebenen MQGET-Aufrufe auf zweite und nachfolgende Anforderungsnachrichten warten, welche mit der durch diese Nachricht gestarteten Arbeitseinheit zusammenhängen. Diese Funktion überschreibt das standardmäßige Warteintervall, das von der Bridge verwendet wird. Sie können die folgenden Sonderwerte verwenden:

MQCGWI_DEFAULT
Standardwarteintervall

Dieser Wert bewirkt, dass die CICS-Brücke für die Dauer wartet, die beim Start der Brücke angegeben wurde.

MQWI_UNLIMITED
Unbegrenztes Warteintervall.

InputItem (MQLONG)
Bei diesem Feld handelt es sich um ein reserviertes Feld. Der Wert muss 0 sein.

Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCIH_VERSION_2 ist.

LinkType (MQLONG)
Dieses Feld ist ein Anforderungsfeld. Sein Anfangswert lautet MQCLT_PROGRAM.

Dieser Wert gibt die Art des Objekts an, mit dem sich die Bridge verbinden möchte. Folgende Werte sind zulässig:

MQCLT_PROGRAM
DPL-Programm.

MQCLT_TRANSACTION
3270-Transaktion.

NextTransactionId (MQCHAR4)
Dieser Wert ist der Name der nächsten Transaktion, die von der Benutzertransaktion (in der Regel von EXEC CICS RETURN TRANSID) zurückgegeben wird. Falls keine weitere Transaktion ansteht, wird dieses Feld auf Leerzeichen gesetzt.

Dieses Feld ist ein Antwortfeld, das nur für 3270-Transaktionen verwendet wird. Die Länge dieses Felds wird durch MQ_TRANSACTION_ID_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

OutputDataLength (MQLONG)
Dieses Feld ist ein Anforderungsfeld, das nur für DPL-Programme verwendet wird. Sein Anfangswert lautet MQCODL_AS_INPUT.

Dieser Wert gibt die Länge der Benutzerdaten an, die in einer Antwortnachricht an den Client zurückgegeben werden sollen. Diese Länge schließt den 8-Byte-Programmnamen ein. Die Länge des Kommunikationsbereichs, der an das verbundene Programm übergeben wird, entspricht dem maximalen Wert dieses Felds und der Länge der Benutzerdaten in der Anforderungsnachricht minus 8.

Anmerkung: Die Länge der Benutzerdaten in einer Nachricht entspricht der Länge der Nachricht ohne MQCIH-Struktur.

Wenn die Länge der Benutzerdaten in der Anforderungsnachricht kleiner als *OutputDataLengt*hist, wird die Option DATALENGTH des Befehls LINK verwendet, sodass LINK effizient an eine andere CICS -Region übertragen werden kann.

Sie können den folgenden Spezialwert verwenden:

MQCODL_AS_INPUT
Ausgabelänge mit Eingabelänge identisch

Dieser Wert ist möglicherweise erforderlich, auch wenn keine Antwort angefordert wird, um sicherzustellen, dass der an das verknüpfte Programm übergebene Kommunikationsbereich eine ausreichende Größe aufweist.

Ursache (MQLONG)
Dieses Feld ist ein Antwortfeld. Sein Anfangswert lautet MQRC_NONE.

Der in diesem Feld zurückgegebene Wert hängt von *ReturnCode* ab; siehe Tabelle 483 auf Seite 283.

RemoteSysId (MQCHAR4)
Dieses Feld enthält die CICS-System-ID des CICS-Systems, von dem die Anforderung verarbeitet wird.

Ist dieses Feld leer, wird die CICS-Systemanforderung in demselben CICS-System verarbeitet wie das Brückenüberwachungsprogramm. Die verwendete SYSID wird in der Antwortnachricht zurückgegeben.

Bei einem 3270-Pseudodialog müssen alle nachfolgenden Nachrichten im Dialog die ferne SYSID angeben, die in der ursprünglichen Antwort zurückgegeben wurde. Falls die SYSID angegeben wird, muss sie folgende Kriterien erfüllen:

- Sie muss aktiv sein.
- Sie muss auf die WebSphere MQ-Anforderungswarteschlange zugreifen können.
- Sie muss für die CICS-ISC-Verbindungen aus dem CICS-System des Brückenüberwachungsprogramms zugänglich sein.

RemoteTransId (MQCHAR4)
Dieses Feld ist ein optionales Anforderungsfeld. Die Länge dieses Felds wird durch MQ_TRANSACTION_ID_LENGTH angegeben.

Erfolgt für dieses Feld eine Angabe, wird es als RTRANSID-Wert von CICS START verwendet.

ReplyToFormat (MQCHAR8)
Der Wert dieses Feldes ist der WebSphere MQ-Formatname der Antwortnachricht, die als Antwort auf die aktuelle Nachricht gesendet wird.

Die Regeln für die Codierung dieses Felds sind mit den Regeln identisch, die für die Codierung des Felds *Format* im MQMD gelten.

Dieses Feld ist ein Anforderungsfeld, das nur für DPL-Programme verwendet wird. Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

Reserved1 (MQCHAR8)
Bei diesem Feld handelt es sich um ein reserviertes Feld. Sein Wert muss aus acht Leerzeichen bestehen.

Reserved2 (MQCHAR8)
Bei diesem Feld handelt es sich um ein reserviertes Feld. Sein Wert muss aus acht Leerzeichen bestehen.

Reserved3 (MQCHAR8)
Bei diesem Feld handelt es sich um ein reserviertes Feld. Sein Wert muss aus acht Leerzeichen bestehen.

Reserved4 (MQLONG)
Bei diesem Feld handelt es sich um ein reserviertes Feld. Der Wert muss 0 sein.

Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCIH_VERSION_2 ist.

ReturnCode (MQLONG)
Der Wert dieses Feldes ist der Rückkehrcode von der CICS-Brücke, der das Ergebnis einer Verarbeitung beschreibt, die von der Brücke ausgeführt wurde. Dieses Feld ist ein Antwortfeld, dessen Anfangswert MQCRC_OK lautet.

Die Felder *Function*, *CompCode*, *Reason* und *AbendCode* enthalten möglicherweise zusätzliche Informationen (siehe Tabelle 483 auf Seite 283). Folgende Werte sind möglich:

MQCRC_APPLICATION_ABEND
(5, X'005') Die Anwendung wurde abnormal beendet.

MQCRC_BRIDGE_ABEND
(4, X'004') CICS-Brücke wurde abnormal beendet

MQCRC_BRIDGE_ERROR
(3, X'003') CICS-Brücke hat einen Fehler erkannt

MQCRC_BRIDGE_TIMEOUT
(8, X'008') Zweite oder spätere Nachricht in der aktuellen Arbeitseinheit nicht innerhalb der angegebenen Zeit empfangen

MQCRC_CICS_EXEC_ERROR
(1, X'001') EXEC CICS-Anweisung hat einen Fehler erkannt

MQCRC_MQ_API_ERROR
(2, X'002') Der MQ-Aufruf hat einen Fehler festgestellt.

MQCRC_OK
(0, X'000') Kein Fehler.

MQCRC_PROGRAM_NOT_AVAILABLE
(7, X'007') Programm nicht verfügbar.

MQCRC_SECURITY_ERROR
(6, X'006') Es ist ein Sicherheitsfehler aufgetreten.

MQCRC_TRANSID_NOT_AVAILABLE
(9, X'009') Transaktion nicht verfügbar.

StartCode (MQCHAR4)
Der Wert dieses Feldes ist ein Indikator, der angibt, ob die Brücke eine Terminaltransaktion oder eine mit START eingeleitete Transaktion emuliert.

Folgende Werte sind möglich:

MQCSC_START
Durchführen des Starts.

MQCSC_STARTDATA
Startdaten.

MQCSC_TERMINPUT
Terminaleingabe.

MQCSC_NONE
Keine.

In allen Fällen sind für die Programmiersprache C die Konstanten MQCSC_*_ARRAY ebenfalls definiert; diese Konstanten enthalten dieselben Werte wie die entsprechenden Konstanten des Typs MQCSC_*, allerdings handelt es sich nicht um Zeichenfolgen, sondern um Feldgruppen von Zeichen.

In der Antwort von der Brücke ist dieses Feld auf den Startcode gesetzt, der der nächsten Transaktions-ID entspricht, die im Feld *NextTransactionId* enthalten ist. Die folgenden Startcodes sind in der Antwort möglich:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT

In CICS Transaction Server Version 1.2 handelt es sich bei diesem Feld um ein reines Anforderungsfeld. Sein Wert in der Antwort ist nicht definiert.

In CICS Transaction Server Version 1.3 und nachfolgenden Releases dient dieses Feld sowohl als Anforderungs- als auch als Antwortfeld.

Dieses Feld wird nur für 3270-Transaktionen verwendet. Die Länge dieses Felds wird durch MQ_START_CODE_LENGTH angegeben. Der Anfangswert dieses Felds lautet MQCSC_NONE.

StrucId (MQCHAR4)
Dieses Feld ist ein Anforderungsfeld, dessen Anfangswert MQCIH_STRUC_ID lautet.

Folgende Werte sind möglich:

MQCIH_STRUC_ID
Die ID für die CICS-Informationsheaderstruktur.

Für die Programmiersprache C ist auch die Konstante MQCIH_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQCIH_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

StrucLength (MQLONG)
Dieses Feld ist ein Anforderungsfeld, dessen Anfangswert MQCIH_LENGTH_2 lautet.

Folgende Werte sind möglich:

MQCIH_LENGTH_1
Die Länge der Version 1 der CICS-Informationsheaderstruktur.

MQCIH_LENGTH_2
Die Länge der Version 2 der CICS-Informationsheaderstruktur.

Die folgende Konstante gibt die Länge der aktuellen Version an:

MQCIH_CURRENT_LENGTH
Die Länge der aktuellen Version der CICS-Informationsheaderstruktur.

TaskEndStatus (MQLONG)
Dieses Feld ist ein Antwortfeld, das den Status der Benutzertransaktion am Ende der Task anzeigt. Dieses Feld, dessen Anfangswert MQCTES_NOSYNC lautet, wird nur für 3270-Transaktionen verwendet.

Einer der folgenden Werte wird zurückgegeben:

MQCTES_NOSYNC
Nicht synchronisiert

Die Benutzertransaktion wurde noch nicht abgeschlossen und weist keinen Synchronisationspunkt auf. Das Feld *MsgType* im MQMD enthält in diesem Fall den Wert MQMT_REQUEST.

MQCTES_COMMIT
Arbeitseinheit festschreiben

Die Benutzertransaktion wurde noch nicht abgeschlossen, aber der ersten Arbeitseinheit wurde ein Synchronisationspunkt zugewiesen. Das Feld *MsgType* im MQMD enthält in diesem Fall den Wert MQMT_DATAGRAM.

MQCTES_BACKOUT
Arbeitseinheit zurücksetzen

Die Benutzertransaktion wurde noch nicht abgeschlossen. Die aktuelle Arbeitseinheit wird zurückgesetzt. Das Feld *MsgType* im MQMD enthält in diesem Fall den Wert MQMT_DATAGRAM.

MQCTES_ENDTASK
Task beenden

Diese Benutzertransaktion wurde beendet (oder abgebrochen). Das Feld *MsgType* im MQMD enthält in diesem Fall den Wert MQMT_REPLY.

TransactionId (MQCHAR4)
Dieses Feld ist ein Anforderungsfeld. Seine Länge wird durch MQ_TRANSACTION_ID_LENGTH angegeben. Der Anfangswert dieses Felds besteht aus vier Leerzeichen.

Wenn für *LinkType* der Wert MQCLT_TRANSACTION angegeben ist, ist *TransactionId* die Transaktions-ID der auszuführenden Benutzertransaktion; belegen Sie in diesem Fall das Feld mit einem Wert.

Wenn für *LinkType* der Wert MQCLT_ROGRAM angegeben ist, ist *TransactionId* der Transaktionscode, unter dem alle Programme innerhalb der Arbeitseinheit ausgeführt werden sollen. Wenn Sie einen Leerwert angeben, wird der Standardtransaktionscode (CKBP) der CICS-DPL-Brücke verwendet. Ist der Wert belegt, müssen Sie ihn für CICS als lokale Transaktion mit einem Startprogramm definiert haben (CSQCBP00). Dieses Feld wird nur benötigt, wenn *UOWControl* den Wert MQCUOWC_FIRST oder MQCUOWC_ONLY aufweist.

UOWControl (MQLONG)
Dieses Feld ist ein Anforderungsfeld, das die Verarbeitung der Arbeitseinheit steuert, die von der CICS-Brücke ausgeführt wird. Der Anfangswert dieses Felds lautet MQCUOWC_ONLY.

Sie können anfordern, dass die Bridge eine einzelne Transaktion oder eines oder mehrere Programme in einer Arbeitseinheit ausführt. Das Feld gibt an, ob die CICS-Brücke eine Arbeitseinheit startet, die angeforderte Funktion innerhalb der aktuellen Arbeitseinheit ausführt oder die Arbeitseinheit beendet, indem diese festgeschrieben oder zurückgesetzt wird. Zur Optimierung der Datenübertragungsabläufe werden verschiedene Kombinationen unterstützt.

Folgende Werte sind möglich:

MQCUOWC_ONLY
Arbeitseinheit starten, Funktion ausführen und anschließend die Arbeitseinheit festschreiben.

MQCUOWC_CONTINUE
Zusatzdaten für die aktuelle Arbeitseinheit (nur bei 3270).

MQCUOWC_FIRST
Arbeitseinheit starten und Funktion ausführen.

MQCUOWC_MIDDLE
Funktion innerhalb der aktuellen Arbeitseinheit ausführen.

MQCUOWC_LAST
Funktion ausführen und anschließend die Arbeitseinheit festschreiben.

MQCUOWC_COMMIT
Arbeitseinheit festschreiben (nur bei DPL).

MQCUOWC_BACKOUT
Arbeitseinheit zurücksetzen (nur bei DPL).

Version (MQLONG)
Dieses Feld ist ein Anforderungsfeld. Sein Anfangswert lautet MQCIH_VERSION_2.

Folgende Werte sind möglich:

MQCIH_VERSION_1
Die Version 1 der CICS-Informationsheaderstruktur.

MQCIH_VERSION_2
Die Version 2 der CICS-Informationsheaderstruktur.

Nur in der neueren Version der Struktur verwendete Felder werden in den Feldbeschreibungen entsprechend gekennzeichnet. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCIH_CURRENT_VERSION
Die aktuelle Version der CICS-Informationsheaderstruktur.

Anfangswerte und Sprachendeklarationen für MQCIH

Tabelle 484. Anfangswerte der Felder in MQCIH für MQCIH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCIH_STRUC_ID	'CIH¬'
Version	MQCIH_VERSION_2	2
StrucLength	MQCIH_LENGTH_2	180
Encoding	--	0
CodedCharSetId	--	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQCIH_NONE	0
ReturnCode	MQCRC_OK	0
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
UOWControl	MQCUOWC_ONLY	273
GetWaitInterval	MQCGWI_DEFAULT	-2
LinkType	MQCLT_PROGRAM	1
OutputDataLength	MQCODL_AS_INPUT	-1
FacilityKeepTime	--	0
ADSDescriptor	MQCADSD_NONE	0
ConversationalTask	MQCCT_NO	0
TaskEndStatus	MQCTES_NOSYNC	0
Facility	MQCFAC_NONE	Nullen
Function	MQCFUNC_NONE	Leerzeichen
AbendCode	--	Leerzeichen
Authenticator	--	Leerzeichen

Tabelle 484. Anfangswerte der Felder in MQCIH für MQCIH (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
Reserved1	--	Leerzeichen
ReplyToFormat	MQFMT_NONE	Leerzeichen
RemoteSysId	--	Leerzeichen
RemoteTransId	--	Leerzeichen
TransactionId	--	Leerzeichen
FacilityLike	--	Leerzeichen
AttentionId	--	Leerzeichen
StartCode	MQCSC_NONE	Leerzeichen
CancelCode	--	Leerzeichen
NextTransactionId	--	Leerzeichen
Reserved2	--	Leerzeichen
Reserved3	--	Leerzeichen
CursorPosition	--	0
ErrorOffset	--	0
InputItem	--	0
Reserved4	--	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQCIH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
  MQCHAR4  StrucId;             /* Structure identifier */
  MQLONG   Version;             /* Structure version number */
  MQLONG   StrucLength;         /* Length of MQCIH structure */
  MQLONG   Encoding;            /* Reserved */
  MQLONG   CodedCharSetId;      /* Reserved */
  MQCHAR8  Format;              /* MQ format name of data that follows
                                   MQCIH */
  MQLONG   Flags;               /* Flags */
  MQLONG   ReturnCode;          /* Return code from bridge */
  MQLONG   CompCode;            /* MQ completion code or CICS EIBRESP */
  MQLONG   Reason;              /* MQ reason or feedback code, or CICS
                                   EIBRESP2 */
  MQLONG   UOWControl;          /* Unit-of-work control */
  MQLONG   GetWaitInterval;     /* Wait interval for MQGET call issued
                                   by bridge task */
  MQLONG   LinkType;            /* Link type */
  MQLONG   OutputDataLength;    /* Output COMMAREA data length */
  MQLONG   FacilityKeepTime;    /* Bridge facility release time */
  MQLONG   ADSDescriptor;       /* Send/receive ADS descriptor */
  MQLONG   ConversationalTask;  /* Whether task can be conversational */
```

```
  MQLONG   TaskEndStatus;       /* Status at end of task */
  MQBYTE8  Facility;            /* Bridge facility token */
  MQCHAR4  Function;            /* MQ call name or CICS EIBFN
                                   function */
  MQCHAR4  AbendCode;           /* Abend code */
  MQCHAR8  Authenticator;       /* Password or passticket */
  MQCHAR8  Reserved1;           /* Reserved */
  MQCHAR8  ReplyToFormat;       /* MQ format name of reply message */
  MQCHAR4  RemoteSysId;         /* Reserved */
  MQCHAR4  RemoteTransId;       /* Reserved */
  MQCHAR4  TransactionId;       /* Transaction to attach */
  MQCHAR4  FacilityLike;        /* Terminal emulated attributes */
  MQCHAR4  AttentionId;         /* AID key */
  MQCHAR4  StartCode;           /* Transaction start code */
  MQCHAR4  CancelCode;          /* Abend transaction code */
  MQCHAR4  NextTransactionId;   /* Next transaction to attach */
  MQCHAR8  Reserved2;           /* Reserved */
  MQCHAR8  Reserved3;           /* Reserved */
  MQLONG   CursorPosition;      /* Cursor position */
  MQLONG   ErrorOffset;         /* Offset of error in message */
  MQLONG   InputItem;           /* Reserved */
  MQLONG   Reserved4;           /* Reserved */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQCIH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Length of MQCIH structure */
  3 Encoding           fixed bin(31), /* Reserved */
  3 CodedCharSetId     fixed bin(31), /* Reserved */
  3 Format             char(8),       /* MQ format name of data that
                                         follows MQCIH */
  3 Flags              fixed bin(31), /* Flags */
  3 ReturnCode         fixed bin(31), /* Return code from bridge */
  3 CompCode           fixed bin(31), /* MQ completion code or CICS
                                         EIBRESP */
  3 Reason             fixed bin(31), /* MQ reason or feedback code, or
                                         CICS EIBRESP2 */
  3 UOWControl         fixed bin(31), /* Unit-of-work control */
  3 GetWaitInterval    fixed bin(31), /* Wait interval for MQGET call
                                         issued by bridge task */
  3 LinkType           fixed bin(31), /* Link type */
  3 OutputDataLength   fixed bin(31), /* Output COMMAREA data length */
  3 FacilityKeepTime   fixed bin(31), /* Bridge facility release time */
  3 ADSDescriptor      fixed bin(31), /* Send/receive ADS descriptor */
  3 ConversationalTask fixed bin(31), /* Whether task can be
                                         conversational */
  3 TaskEndStatus      fixed bin(31), /* Status at end of task */
  3 Facility           char(8),       /* Bridge facility token */
  3 Function           char(4),       /* MQ call name or CICS EIBFN
                                         function */
  3 AbendCode          char(4),       /* Abend code */
  3 Authenticator      char(8),       /* Password or passticket */
  3 Reserved1          char(8),       /* Reserved */
  3 ReplyToFormat      char(8),       /* MQ format name of reply
                                         message */
  3 RemoteSysId        char(4),       /* Reserved */
  3 RemoteTransId      char(4),       /* Reserved */
  3 TransactionId      char(4),       /* Transaction to attach */
  3 FacilityLike       char(4),       /* Terminal emulated attributes */
  3 AttentionId        char(4),       /* AID key */
  3 StartCode          char(4),       /* Transaction start code */
  3 CancelCode         char(4),       /* Abend transaction code */
  3 NextTransactionId  char(4),       /* Next transaction to attach */
  3 Reserved2          char(8),       /* Reserved */
  3 Reserved3          char(8),       /* Reserved */
  3 CursorPosition     fixed bin(31), /* Cursor position */
  3 ErrorOffset        fixed bin(31), /* Offset of error in message */
  3 InputItem          fixed bin(31), /* Reserved */
  3 Reserved4          fixed bin(31); /* Reserved */
```

Deklaration in High Level Assembler

```
MQCIH                     DSECT
MQCIH_STRUCID             DS   CL4  Structure identifier
MQCIH_VERSION             DS   F    Structure version number
MQCIH_STRUCLENGTH         DS   F    Length of MQCIH structure
MQCIH_ENCODING            DS   F    Reserved
MQCIH_CODEDCHARSETID      DS   F    Reserved
MQCIH_FORMAT              DS   CL8  MQ format name of data that follows
*                                   MQCIH
MQCIH_FLAGS               DS   F    Flags
MQCIH_RETURNCODE          DS   F    Return code from bridge
MQCIH_COMPCODE            DS   F    MQ completion code or CICS EIBRESP
MQCIH_REASON              DS   F    MQ reason or feedback code, or CICS
*                                   EIBRESP2
MQCIH_UOWCONTROL          DS   F    Unit-of-work control
MQCIH_GETWAITINTERVAL     DS   F    Wait interval for MQGET call issued
*                                   by bridge task
MQCIH_LINKTYPE            DS   F    Link type
MQCIH_OUTPUTDATALENGTH    DS   F    Output COMMAREA data length
MQCIH_FACILITYKEEPTIME    DS   F    Bridge facility release time
MQCIH_ADSDESCRIPTOR       DS   F    Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK  DS   F    Whether task can be conversational
MQCIH_TASKENDSTATUS       DS   F    Status at end of task
MQCIH_FACILITY            DS   XL8  Bridge facility token
MQCIH_FUNCTION            DS   CL4  MQ call name or CICS EIBFN function
MQCIH_ABENDCODE           DS   CL4  Abend code
MQCIH_AUTHENTICATOR       DS   CL8  Password or passticket
MQCIH_RESERVED1           DS   CL8  Reserved
MQCIH_REPLYTOFORMAT       DS   CL8  MQ format name of reply message
MQCIH_REMOTESYSID         DS   CL4  Reserved
MQCIH_REMOTETRANSID       DS   CL4  Reserved
MQCIH_TRANSACTIONID       DS   CL4  Transaction to attach
MQCIH_FACILITYLIKE        DS   CL4  Terminal emulated attributes
MQCIH_ATTENTIONID         DS   CL4  AID key
MQCIH_STARTCODE           DS   CL4  Transaction start code
MQCIH_CANCELCODE          DS   CL4  Abend transaction code
MQCIH_NEXTTRANSACTIONID   DS   CL4  Next transaction to attach
MQCIH_RESERVED2           DS   CL8  Reserved
MQCIH_RESERVED3           DS   CL8  Reserved
MQCIH_CURSORPOSITION      DS   F    Cursor position
MQCIH_ERROROFFSET         DS   F    Offset of error in message
MQCIH_INPUTITEM           DS   F    Reserved
MQCIH_RESERVED4           DS   F    Reserved
*
MQCIH_LENGTH              EQU  *-MQCIH
                          ORG  MQCIH
MQCIH_AREA                DS   CL(MQCIH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQCIH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Length of MQCIH structure'
  Encoding           As Long     'Reserved'
  CodedCharSetId     As Long     'Reserved'
  Format             As String*8 'MQ format name of data that follows'
                                 'MQCIH'
  Flags              As Long     'Flags'
  ReturnCode         As Long     'Return code from bridge'
  CompCode           As Long     'MQ completion code or CICS EIBRESP'
  Reason             As Long     'MQ reason or feedback code, or CICS'
                                 'EIBRESP2'
  UOWControl         As Long     'Unit-of-work control'
  GetWaitInterval    As Long     'Wait interval for MQGET call issued'
                                 'by bridge task'
  LinkType           As Long     'Link type'
  OutputDataLength   As Long     'Output COMMAREA data length'
  FacilityKeepTime   As Long     'Bridge facility release time'
  ADSDescriptor      As Long     'Send/receive ADS descriptor'
  ConversationalTask As Long     'Whether task can be conversational'
  TaskEndStatus      As Long     'Status at end of task'
  Facility           As MQBYTE8  'Bridge facility token'
  Function           As String*4 'MQ call name or CICS EIBFN function'
  AbendCode          As String*4 'Abend code'
  Authenticator      As String*8 'Password or passticket'
  Reserved1          As String*8 'Reserved'
  ReplyToFormat      As String*8 'MQ format name of reply message'
```

```
   RemoteSysId        As String*4 'Reserved'
   RemoteTransId      As String*4 'Reserved'
   TransactionId      As String*4 'Transaction to attach'
   FacilityLike       As String*4 'Terminal emulated attributes'
   AttentionId        As String*4 'AID key'
   StartCode          As String*4 'Transaction start code'
   CancelCode         As String*4 'Abend transaction code'
   NextTransactionId  As String*4 'Next transaction to attach'
   Reserved2          As String*8 'Reserved'
   Reserved3          As String*8 'Reserved'
   CursorPosition     As Long     'Cursor position'
   ErrorOffset        As Long     'Offset of error in message'
   InputItem          As Long     'Reserved'
   Reserved4          As Long     'Reserved'
End Type
```

MQCMHO - Optionen zum Erstellen von Nachrichtenkennungen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 485. Felder in MQCMHO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options

Überblick über MQCMHO

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS und WebSphere MQ -Clients.

Zweck: Mit der **MQCMHO**-Struktur können Anwendungen Optionen festlegen, die steuern, wie Nachrichtenkennungen erstellt werden. Die Struktur ist ein Eingabeparameter für den **MQCRTMH**-Aufruf.

Zeichensatz und Codierung: Die Daten in **MQCMHO** müssen im Zeichensatz und in der Codierung der Anwendung (**MQENC_NATIVE**) enthalten sein.

Felder für MQCMHO

Die MQCMHO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert ist MQCMHO_DEFAULT_VALIDATION.

Eine der folgenden Optionen kann angegeben werden:

MQCMHO_VALIDATE

Wird **MQSETMP** aufgerufen, um eine Eigenschaft in dieser Nachrichtenkennung zu definieren, wird sichergestellt, dass der Eigenschaftsname folgende Bedingungen erfüllt:

- keine ungültigen Zeichen enthält.
- Er beginnt nicht mit "JMS" oder "usr.JMS", ausgenommen in folgenden Fällen:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Diese Namen sind für JMS-Eigenschaften reserviert.

- Es handelt sich nicht um eines der folgenden Schlüsselwörter in einer beliebigen Mischung von Groß- und Kleinbuchstaben:
  - AND
  - ZWISCHEN
  - Esc
  - FALSE
  - IN
  - IST
  - LIKE
  - NICHT
  - NULL
  - OR
  - TRUE
- Er beginnt nicht mit 'Body.' oder 'Root.' (außer Root.MQMD.)

Wenn die Eigenschaft MQ-definiert ist (mq. *) Wenn der Name erkannt wird, werden die Eigenschaftsdeskriptorfelder auf die richtigen Werte für die Eigenschaft gesetzt. Wird die Eigenschaft nicht erkannt, wird das Feld *Support* des Eigenschaftsdeskriptors auf **MQPD_OPTIONAL** gesetzt.

MQCMHO_DEFAULT_VALIDATION

Dieser Wert gibt an, dass die Standardüberprüfungsstufe für Eigenschaftsnamen gilt.

Die Standardüberprüfungsstufe entspricht der von **MQCMHO_VALIDATE** festgelegten Stufe.

Dies ist der Standardwert.

MQCMHO_NO_VALIDATION

Am Eigenschaftsnamen wird keine Überprüfung vorgenommen. Siehe Beschreibung von **MQCMHO_VALIDATE**.

Standardoption: Wenn keine der oben beschriebenen Optionen erforderlich ist, kann folgende Option verwendet werden:

MQCMHO_NONE

Alle Optionen nehmen ihren Standardwert an. Verwenden Sie diesen Wert, um anzugeben, dass keine anderen Optionen angegeben wurden. **MQCMHO_NONE** unterstützt die Programmdokumentation. Diese Option soll mit keiner anderen Option verwendet werden, aber da ihr Wert Null ist, kann ihre Verwendung nicht erkannt werden.

StrucId (MQCHAR4)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert ist MQCMHO_STRUC_ID.

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQCMHO_STRUC_ID
ID für die Optionsstruktur zur Erstellung von Nachrichtenkennungen.

Für die Programmiersprache C wird außerdem die Konstante **MQCMHO_STRUC_ID_ARRAY** definiert; diese hat denselben Wert wie **MQCMHO_STRUC_ID**, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Version (MQLONG)
Dieses Feld ist immer ein Eingabefeld. Sein Anfangswert ist MQCMHO_VERSION_1.

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQCMHO_VERSION_1
Version-1 der Struktur für die Erstellung von Optionen für Nachrichtenkennungen.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCMHO_CURRENT_VERSION
Aktuelle Version der Struktur für die Erstellung von Optionen für Nachrichtenkennungen.

Anfangswerte und Sprachendeklarationen für MQCMHO

Tabelle 486. Anfangswerte der Felder in MQCMHO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCMHO_STRUC_ID	'CMHO'
Version	MQCMHO_VERSION_1	1
Options	MQCMHO_DEFAULT_VALIDATION	0

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQCMHO_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Deklaration in Programmiersprache C

```
struct tagMQCMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQCRTMH */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQCMHO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action of MQCRTMH */
```

Deklaration in High Level Assembler

```
MQCMHO                DSECT
MQCMHO_STRUCID        DS   CL4   Structure identifier
MQCMHO_VERSION        DS   F     Structure version number
MQCMHO_OPTIONS        DS   F     Options that control the action of
*                                MQCRTMH
```

```
MQCMHO_LENGTH         EQU  *-MQCMHO
MQCMHO_AREA           DS   CL(MQCMHO_LENGTH)
```

MQCNO - Verbindungsoptionen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 487. Felder in MQCNO		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQCONNX	Optionen
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQCNO_VERSION_2 ist.		
ClientConnOffset	Relative Position der MQCD-Struktur für die Clientverbindung	ClientConnOffset
ClientConnPtr	Adresse der MQCD-Struktur für die Clientverbindung	ClientConnPtr
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQCNO_VERSION_3 ist.		
ConnTag	Verbindungstag des Warteschlangenmanagers	ConnTag
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQCNO_VERSION_4 ist.		
SSLConfigPtr	Adresse der MQSCO-Struktur für die Clientverbindung	SSLConfigPtr
SSLConfigOffset	Relative Position der MQSCO-Struktur für die Clientverbindung	SSLConfigOffset
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQCNO_VERSION_5 ist.		
ConnectionId	Eindeutige Verbindungs-ID	ConnectionId
SecurityParmsOffset	Sicherheitsparameter	SecurityParmsOffset
SecurityParmsPtr	Sicherheitsparameter	SecurityParmsPtr

Zugehörige Tasks

MQCONNX verwenden

Übersicht über MQCNO

Verfügbarkeit: Alle Versionen mit Ausnahme von MQCNO_VERSION_4: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQCNO-Struktur ermöglicht der Anwendung die Angabe von Optionen für die Verbindung mit dem lokalen Warteschlangenmanager. Die Struktur ist ein Ein-/Ausgabeparameter im MQCONNX-Anruf. Weitere Informationen zur Verwendung gemeinsamer Kennungen und zum MQCONNX-Aufruf finden Sie im Abschnitt Gemeinsam genutzte (threadunabhängige) Verbindungen mit MQCONNX.

Version: Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützte Programmiersprachen bereitgestellt werden, erhalten die aktuellste Version von MQCNO, wobei der Anfangswert des Feldes *Version* auf MQCNO_VERSION_1 gesetzt ist. Um Felder zu verwenden, die in der Struktur der Version 1 nicht verfügbar sind, muss die Anwendung das Feld *Version* auf die Versionsnummer der erforderlichen Version setzen.

Zeichensatz und Codierung: Die Daten in MQCNO müssen den Zeichensatz haben, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und die Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE festgelegt wird. Wird die Anwendung jedoch als WebSphere MQ-MQI-Client ausgeführt, muss die Struktur den Zeichensatz und die Codierung des Clients aufweisen.

Felder für MQCNO

Die MQCNO-Struktur enthält die folgenden Felder, die in **alphabetischer Reihenfolge** beschrieben werden:

ClientConnOffset (MQLONG)
ClientConnOffset gibt die relative Position in Bytes einer MQCD-Kanaldefinitionsstruktur ab Beginn der MQCNO-Struktur an. Der Offset kann positiv oder negativ sein. Dieses Feld ist ein Eingabefeld, dessen Anfangswert 0 ist.

Verwenden Sie das Feld *ClientConnOffset* nur, wenn die Anwendung, die den MQCONNX-Aufruf ausgibt, als WebSphere MQ-MQI-Client ausgeführt wird. Sie finden Informationen zur Verwendung dieses Felds in der Beschreibung des Felds *ClientConnPtr*.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_2 ist.

ClientConnPtr (MQPTR)
ClientConnPtr ist ein Eingabefeld. Sein Anfangswert ist bei Programmiersprachen, die Zeiger unterstützen, ein Nullzeiger. Andernfalls ist der Anfangswert eine Bytefolge, die gänzlich aus Nullen besteht.

Verwenden Sie die Felder *ClientConnOffset* und *ClientConnPtr* nur, wenn die Anwendung, die den MQCONNX-Aufruf ausgibt, als WebSphere MQ-MQI-Client ausgeführt wird. Durch die Angabe eines oder mehrerer dieser Felder kann die Anwendung die Definition des Clientverbindungskanals steuern, indem sie eine MQCD-Kanaldefinitionsstruktur bereitstellt, die die erforderlichen Werte enthält.

Wird die Anwendung als WebSphere MQ-MQI-Client ausgeführt und stellt jedoch keine MQCD-Struktur bereit, wird die Kanaldefinition mit der Umgebungsvariablen MQSERVER ausgewählt. Wurde MQSERVER nicht festgelegt, wird die Clientkanaltabelle verwendet.

Wenn die Anwendung nicht als WebSphere MQ-MQI-Client ausgeführt wird, werden die Felder *ClientConnOffset* und *ClientConnPtr* ignoriert.

Wenn die Anwendung eine MQCD-Struktur bereitstellt, legen Sie in den aufgeführten Feldern die erforderlichen Werte fest. Die übrigen Felder in MQCD werden ignoriert. Sie können Zeichenfolgen mit Leerzeichen bis zur Länge des Felds auffüllen oder diese mit einem Nullzeichen beenden. Im Abschnitt „Felder” auf Seite 1060 finden Sie weitere Informationen zu den Feldern in der MQCD-Struktur.

Feld in MQCD	**Wert**
ChannelName	Der Kanalname.
Version	Strukturversionsnummer. Sie darf nicht kleiner als MQCD_VERSION_7 sein.
TransportType	Ein beliebiger unterstützter Transporttyp.
ModeName	Der Name des LU 6.2-Modus.
TpName	Gibt das LU 6.2-Transaktionsprogramm an.
SecurityExit	Der Name des Kanalsicherheitsexits.
SendExit	Der Name des Kanalsendeexits.
ReceiveExit	Der Name des Kanalempfangsexits.
MaxMsgLength	Die maximale Länge (in Bytes) von Nachrichten, die über den Clientverbindungskanal gesendet werden können.
SecurityUserData	Die Benutzerdaten für den Sicherheitsexit.

Feld in MQCD	Wert
SendUserData	Die Benutzerdaten für den Sendeexit.
ReceiveUserData	Die Benutzerdaten für den Empfangsexit.
UserIdentifier	Die Benutzer-ID, die für den Aufbau einer LU 6.2-Sitzung verwendet werden soll.
Password	Das Kennwort, das für den Aufbau einer LU 6.2-Sitzung verwendet werden soll.
ConnectionName	Gibt den Namen der Verbindung an.
HeartbeatInterval	Die Zeit in Sekunden, die zwischen dem Austausch von Überwachungssignalen liegt.
StrucLength	Die Länge der MQCD-Struktur.
ExitNameLength	Die Länge der Exitnamen, die durch *SendExitPtr* und *ReceiveExitPtr* adressiert werden. Sie muss größer als null sein, wenn *SendExitPtr* oder *ReceiveExitPtr* auf einen Wert gesetzt ist, der kein Nullzeiger ist.
ExitDataLength	Die Länge der Exitdaten, die durch *SendUserDataPtr* und *ReceiveUserDataPtr* adressiert werden. Sie muss größer als null sein, wenn *SendUserDataPtr* oder *ReceiveUserDataPtr* auf einen Wert gesetzt ist, der kein Nullzeiger ist.
SendExitsDefined	Die Anzahl der Sendeexits, die durch *SendExitPtr* adressiert werden. Beim Wert null werden der Exitname und die Daten durch *SendExit* und *SendUserData* bereitgestellt. Ist der Wert größer als null, werden die Exitnamen und Daten durch *SendExitPtr* und *SendUserDataPtr* bereitgestellt, und *SendExit* und *SendUserData* müssen leer sein.
ReceiveExitsDefined	Die Anzahl der Empfangsexits, die durch *ReceiveExitPtr* adressiert werden. Beim Wert null werden der Exitname und die Daten durch *ReceiveExit* und *ReceiveUserData* bereitgestellt. Ist der Wert größer als null, werden die Exitnamen und Daten durch *ReceiveExitPtr* und *ReceiveUserDataPtr* bereitgestellt, und *ReceiveExit* und *ReceiveUserData* müssen leer sein.
SendExitPtr	Die Adresse des ersten Sendeexitnamens.
SendUserDataPtr	Die Adresse der Daten für den ersten Sendeexit.
ReceiveExitPtr	Die Adresse des ersten Empfangsexitnamens.
ReceiveUserDataPtr	Die Adresse der Daten für den ersten Empfangsexit.
LongRemoteUserIdLength	Die Länge der langen ID des fernen Benutzers.
LongRemoteUserIdPtr	Die Adresse der langen ID des fernen Benutzers.
RemoteSecurityId	Die ferne Sicherheits-ID.
SSLCipherSpec	Die SSL-CipherSpec.
SSLPeerNamePtr	Die Adresse des SSL-Peernamens.
SSLPeerNameLength	Die Länge des SSL-Peernamens.
KeepAliveInterval	Der an den Kommunikationsstack übergebene Wert, der das Keepalive-Timing für den Kanal festlegt.
LocalAddress	Die lokale Kommunikationsadresse einschließlich der IP-Adresse des zu verwendenden lokalen Netzadapters sowie eines Portbereichs, der für abgehende Verbindungen verwendet wird.

Nutzen Sie für die Bereitstellung der Kanaldefinitionsstruktur eines der folgenden beiden Verfahren:

- Verwendung des relativen Positionsfelds *ClientConnOffset*

  In diesem Fall muss die Anwendung eine zusammengesetzte Struktur deklarieren, die eine MQCNO-Struktur enthält, der die Kanaldefinitionsstruktur MQCD folgt. Sie muss außerdem *ClientConnOffset* auf die relative Position der Kanaldefinitionsstruktur ab Beginn der MQCNO-Struktur setzen. Vergewissern Sie sich, dass diese relative Position korrekt ist. *ClientConnPtr* muss auf den Nullzeiger oder auf Nullbytes gesetzt werden.

  Verwenden Sie *ClientConnOffset* für Programmiersprachen, die den Zeigerdatentyp nicht unterstützen oder diesen auf eine Weise implementieren, die nicht auf verschiedene Umgebungen übertragen werden kann (beispielsweise bei der Programmiersprache COBOL).

  Für die Programmiersprache Visual Basic wird eine zusammengesetzte Struktur namensMQCNOCD in der Headerdatei CMQXB.BAS bereitgestellt; diese Struktur enthält eine MQCNO-Struktur, auf die eine MQCD-Struktur folgt. Initialisieren Sie MQCNOCD, indem Sie die Subroutine MQCNOCD_DEFAULTS aufrufen. MQCNOCD wird in Verbindung mit der VarianteMQCONNXAny-Variante des MQCONNX-Aufrufs. Die Beschreibung des MQCONNX-Aufrufs enthält nähere Informationen hierzu.

- Verwendung des Zeigerfelds *ClientConnPtr*

  In diesem Fall kann die Anwendung die Kanaldefinitionsstruktur gesondert von der MQCNO-Struktur deklarieren und *ClientConnPtr* auf die Adresse der Kanaldefinitionsstruktur setzen. Legen Sie für *ClientConnOffset* den Wert null fest.

  Verwenden Sie *ClientConnPtr* bei Programmiersprachen, die den Zeigerdatentyp auf eine Weise unterstützen, die auf verschiedene Umgebungen übertragbar ist (beispielsweise bei der Programmiersprache C).

  In der Programmiersprache C können Sie die Makrovariable MQCD_CLIENT_CONN_DEFAULT zur Bereitstellung von Anfangswerten für die Struktur verwenden, die für die Nutzung im MQCONNX-Aufruf besser geeignet sind als die Anfangswerte, die durch MQCD_DEFAULT bereitgestellt werden.

Ungeachtet des gewählten Verfahrens darf nur entweder *ClientConnOffset* oder *ClientConnPtr* verwendet werden; haben beide Felder einen Wert ungleich null, schlägt der Aufruf mit dem Ursachencode MQRC_CLIENT_CONN_ERROR fehl.

Sobald der MQCONNX-Aufruf abgeschlossen ist, wird nicht mehr auf die MQCD-Struktur verwiesen.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_2 ist.

Anmerkung: Bei Plattformen, bei denen die Programmiersprache den Datentyp Zeiger nicht unterstützt, wird das Feld als Bytefolge der entsprechenden Länge deklariert, wobei der Anfangswert die vollständig auf null gesetzte Bytefolge ist.

ConnectionId (MQBYTE24)
ConnectionId ist eine eindeutige ID mit 24 Bytes, die WebSphere MQ eine verlässliche Identifizierung einer Anwendung ermöglicht. Eine Anwendung kann diese ID für die Korrelation in PUT- und GET-Aufrufen verwenden. Dieser Ausgabeparameter hat in allen Programmiersprachen einen Anfangswert von 24 Nullbytes.

Der Warteschlangenmanager ordnet allen Verbindungen eine eindeutige ID zu, und zwar ungeachtet dessen, wie sie eingerichtet wurden. Wenn ein MQCONNX-Aufruf die Verbindung mit einer MQCNO-Struktur der Version 5 einrichtet, kann die Anwendung die Verbindungs-ID (ConnectionId) über den zurückgegebenen MQCNO-Wert ermitteln. Die Eindeutigkeit der zugeordneten ID unter allen anderen IDs (beispielsweise CorrelId, MsgID und GroupId), die von WebSphere MQ generiert werden, wird garantiert.

Verwenden Sie das Feld "ConnectionId" für die Ermittlung von Arbeitseinheiten mit langer Ausführungszeit, indem Sie den PCF-Befehl zur Abfrage der Verbindung (Inquire Connection) oder den WebSphere MQ-Scriptbefehl DISPLAY CONN ausgeben. Die von WebSphere MQ-Scriptbefehlen (CONN) verwendete Verbindungs-ID wird von der hier zurückgegebenen Verbindungs-ID abgeleitet. Die PCF-Befehle "Inquire"

und "Stop Connection" für die Abfrage bzw. das Stoppen einer Verbindung können die hier zurückgegebene Verbindungs-ID unverändert verwenden.

Über "ConnectionId" kann die Beendigung einer Arbeitseinheit mit langer Ausführungszeit erzwungen werden. Hierfür muss die Verbindungs-ID (ConnectionId) unter Verwendung des PCF-Befehls "Stop Connection" oder des WebSphere MQ-Scriptbefehls STOP CONN angegeben werden. Weitere Informationen zur Verwendung dieser Befehle finden Sie in den Abschnitten Stop Connection und STOP CONN.

Dieses Feld wird nicht zurückgegeben, wenn die Version kleiner als MQCNO_VERSION_5 ist.

Die Länge dieses Felds wird durch MQ_CONNECTION_ID_LENGTH angegeben.

ConnTag (MQBYTE128)
Das Feld "ConnTag" gibt einen Tag an, den der Warteschlangenmanager den Ressourcen zuordnet, die während der Verbindung durch die Anwendung beeinflusst werden. Jede Anwendung oder Anwendungsinstanz muss einen anderen Wert für diesen Tag verwenden, damit der Warteschlangenmanager den Zugriff auf die betroffenen Ressourcen korrekt serialisieren kann. Dieses Feld ist ein Eingabefeld, dessen Anfangswert MQCT_NONE lautet.

In den Beschreibungen der MQCNO_*-Optionen für _CONN_TAG_* finden Sie nähere Informationen zu den Werten, die von den verschiedenen Anwendungen verwendet werden müssen. Der Tag verliert seine Gültigkeit, sobald die Anwendung beendet wird oder den Aufruf MQDISC ausgibt.

Anmerkung: Werte für Verbindungstags, die mit MQ in Groß- und/oder Kleinschreibung beginnen und in ASCII- oder EBCDIC-Code geschrieben werden, sind für die Verwendung durch IBM Produkte reserviert. Verwenden Sie also keine Werte für Verbindungstags, die mit diesen Buchstaben beginnen.

Wenn Sie keinen Tag benötigen, verwenden Sie folgenden Sonderwert:

MQCT_NONE
Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQCT_NONE_ARRAY definiert; diese Konstante hat denselben Wert wie die Konstante MQCT_NONE. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Dieses Feld wird verwendet, wenn eine Verbindung zu einem z/OS-Warteschlangenmanager hergestellt wird. Geben Sie in anderen Umgebungen den Wert MQCT_NONE an.

Die Länge dieses Felds wird durch MQ_CONN_TAG_LENGTH angegeben. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_3 ist.

Optionen (MQLONG)
Optionen, mit denen die Aktion des MQCONNX-Aufrufs gesteuert wird.

Abrechnungsoptionen

Die folgenden Optionen steuern die Art der Abrechnung, wenn das Warteschlangenmanagerattribut *AccountingConnOverride* auf MQMON_ENABLED gesetzt ist:

MQCNO_ACCOUNTING_MQI_ENABLED
Wenn die Erfassung von Überwachungsdaten in der Warteschlangenmanagerdefinition inaktiviert wird, indem das Attribut *MQIAccounting* auf MQMON_OFF gesetzt wird, wird die Erfassung von MQI-Abrechnungsdaten aktiviert.

MQCNO_ACCOUNTING_MQI_DISABLED
Wenn die Erfassung von Überwachungsdaten in der Warteschlangenmanagerdefinition inaktiviert wird, indem das Attribut *MQIAccounting* auf MQMON_OFF gesetzt wird, wird die Erfassung von MQI-Abrechnungsdaten gestoppt.

MQCNO_ACCOUNTING_Q_ENABLED
Wenn die Erfassung von Warteschlangenabrechnungsdaten in der Warteschlangenmanagerdefinition inaktiviert wird, indem das *MQIAccounting* -Attribut auf MQMON_OFF gesetzt wird, aktiviert die

Einstellung dieses Flags die Erfassung von Abrechnungsdaten für die Warteschlangen, die einen Warteschlangenmanager im Feld *MQIAccounting* ihrer Warteschlangendefinition angeben.

MQCNO_ACCOUNTING_Q_DISABLED
Wenn die Erfassung von Warteschlangenabrechnungsdaten in der Warteschlangenmanagerdefinition inaktiviert wird, indem das Attribut *MQIAccounting* auf MQMON_OFF gesetzt wird, wird die Erfassung von Abrechnungsdaten für die Warteschlangen, die einen Warteschlangenmanager im Feld *MQIAccounting* ihrer Warteschlangendefinition angeben, inaktiviert.

Wird keine dieser Optionen angegeben, erfolgt die Abrechnung für die Verbindung wie in den Warteschlangenmanagerattributen definiert.

Bindungsoptionen

Mit den folgenden Optionen wird die WebSphere MQ-Bindung festgelegt, die verwendet werden soll. Es darf nur jeweils eine dieser Optionen angegeben werden:

MQCNO_STANDARD_BINDING
Die Anwendung und der lokale Warteschlangenmanageragent (die Komponente, die die Einreihungsoperationen steuert) sind jeweils in eigenen Ausführungseinheiten (normalerweise in separaten Prozessen) aktiv. Dadurch wird die Integrität des Warteschlangenmanagers aufrechterhalten, d. h., er wird vor fehlgeleiteten Programmen geschützt.

Unterstützt der Warteschlangenmanager mehrere Bindungstypen und wird MQCNO_STANDARD_BINDING gesetzt, wählt der Warteschlangenmanager über das Attribut *DefaultBindType* in der Zeilengruppe *Connection* der Datei `qm.ini` (oder im entsprechenden Windows-Registrierungseintrag) den Bindungstyp aus. Ist diese Zeilengruppe nicht definiert oder kann der Wert nicht verwendet oder nicht für die Anwendung verwendet werden, wählt der Warteschlangenmanager den entsprechenden Bindungstyp aus. Der Warteschlangenmanager setzt den in den Verbindungsoptionen verwendeten Bindungstyp.

Die Option MQCNO_STANDARD_BINDING wird verwendet, wenn die Anwendung noch nicht vollständig getestet wurde oder störanfällig oder nicht vertrauenswürdig ist. MQCNO_STANDARD_BINDING ist der Standardwert.

Diese Option wird in allen Umgebungen unterstützt.

Wenn Sie eine Verbindung zur Bibliothek `mqm` herstellen, wird zunächst versucht, eine standardmäßige Serververbindung unter Verwendung des Standardbindungstyps herzustellen. Kann die entsprechende Serverbibliothek nicht geladen werden, wird versucht, eine Clientverbindung herzustellen.

- Bei Angabe der Umgebungsvariablen MQ_CONNECT_TYPE kann eine der folgenden Optionen verwendet werden, um das Verhalten von MQCONN oder MQCONNX zu ändern, wenn MQCNO_STANDARD_BINDING angegeben wurde (außer wenn MQCNO_FASTPATH_BINDING angegeben wird und MQ_CONNECT_TYPE auf LOCAL oder STANDARD gesetzt ist, damit der Administrator für Fastpath-Verbindungen einen Downgrade durchführen kann, ohne dass Änderungen an der Anwendung vorgenommen werden müssen):

Wert	Bedeutet
CLIENT	Es wird nur versucht, eine Clientverbindung herzustellen.
FASTPATH	Dieser Wert wird in früheren Releases unterstützt, jetzt bei Angabe aber ignoriert.
LOKAL	Es wird versucht, nur eine Serververbindung herzustellen. Für Fastpath-Verbindungen wird ein Downgrade auf eine Standardserververbindung durchgeführt.

Wert	Bedeutet
STANDARD	Wird aus Gründen der Kompatibilität mit früheren Releases unterstützt. Dieser Wert wird nicht wie LOCAL gehandhabt.

- Ist die Umgebungsvariable MQ_CONNECT_TYPE beim Aufruf von MQCONNX nicht gesetzt, wird versucht, unter Verwendung des Standardbindungstyps eine Standardserverbindung herzustellen. Kann die Serverbibliothek nicht geladen werden, wird versucht, eine Clientverbindung herzustellen.

MQCNO_FASTPATH_BINDING

Die Anwendung und der lokale Warteschlangenmanageragent sind Teil derselben Ausführungseinheit. Dies unterscheidet sich vom typischen Bindungsverfahren, bei dem Anwendung und lokaler Warteschlangenmanageragent jeweils in eigenen Ausführungseinheiten aktiv sind.

MQCNO_FASTPATH_BINDING wird ignoriert, wenn dieser Bindungstyp vom Warteschlangenmanager nicht unterstützt wird; die Verarbeitung wird so fortgesetzt, als ob diese Option nicht angegeben wurde.

MQCNO_FASTPATH_BINDING kann von Vorteil sein, wenn eine Vielzahl von Prozessen mehr als die von der Anwendung insgesamt verwendeten Ressourcen verbrauchen.Eine Anwendung, die die Fastpath-Bindung verwendet, wird als *vertrauenswürdige Anwendung* bezeichnet.

Bei der Entscheidung, ob die Fastpath-Bindung verwendet werden soll, sollten Sie Folgendes bedenken:

- Mit der Option MQCNO_FASTPATH_BINDING kann nicht verhindert werden, dass eine Anwendung Nachrichten und andere Datenbereiche, die zum Warteschlangenmanager gehören, ändert oder zerstört. Diese Option sollte nur verwendet werden, wenn Sie sich über die Konsequenzen im Klaren sind.
- Die Anwendung darf keine asynchronen Signale oder Zeitgeberinterrupts (wie `sigkill`) zusammen mit MQCNO_FASTPATH_BINDING verwenden. Darüber hinaus bestehen auch Einschränkungen hinsichtlich der Verwendung gemeinsam genutzter Speichersegmente.
- Die Anwendung muss die Verbindung zum Warteschlangenmanager mithilfe des MQDISC-Aufrufs beenden.
- Die Anwendung muss beendet werden, bevor der Warteschlangenmanager mit dem Befehl `endmqm` beendet wird.
- Unter IBM i muss der Job unter einem Benutzerprofil ausgeführt werden, das zur Gruppe `QMQMADM` gehört. Darüber hinaus darf das Programm nicht abnormal beendet werden, da dies zu unvorhersehbaren Ergebnissen führen kann.
- Auf UNIX -Systemen muss die `mqm` -Benutzer-ID die effektive Benutzer-ID und die `mqm` -Gruppen-ID die effektive Gruppen-ID sein. Damit die Anwendung auf diese Weise ausgeführt wird, konfigurieren Sie das Programm so, dass es der Benutzer-ID `mqm` und der Gruppen-ID `mqm` gehört, und legen Sie dann die Berechtigungsbits `setuid` und `setgid` für das Programm fest.

  WebSphere MQ Object Authority Manager (OAM) verwendet für die Berechtigungsprüfung nach wie vor die reale Benutzer-ID:
- Unter Windows muss das Programm zur Gruppe `mqm` gehören. Für 64-Bit-Anwendungen wird die Fastpath-Bindung nicht unterstützt.

Die Option MQCNO_FASTPATH_BINDING wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linuxund Windows. Unter z/OS wird sie zwar akzeptiert, jedoch ignoriert.

Weitere Informationen zu den Auswirkungen der Verwendung vertrauenswürdiger Anwendungen finden Sie unter Einschränkungen für vertrauenswürdige Anwendungen .

MQCNO_SHARED_BINDING

Mit MQCNO_SHARED_BINDING nutzen die Anwendung und der lokale WS-Manager-Agent einige Ressourcen gemeinsam. Wenn der Warteschlangenmanager diesen Bindungstyp nicht unterstützt, wird

MQCNO_SHARED_BINDING ignoriert. Die Verarbeitung wird so fortgesetzt, als ob diese Option nicht angegeben wurde.

MQCNO_ISOLATED_BINDING

In diesem Fall sind Anwendungsprozess und lokaler Warteschlangenmanageragent separat aktiv; sie nutzen keine Ressourcen gemeinsam. Wenn der Warteschlangenmanager diesen Bindungstyp nicht unterstützt, wird MQCNO_ISOLATED_BINDING ignoriert. Die Verarbeitung wird so fortgesetzt, als ob diese Option nicht angegeben wurde.

MQCNO_CLIENT_BINDING

Geben Sie diese Option an, wenn die Anwendung nur eine Clientverbindung herstellen soll. Für diese Option gelten die folgenden Einschränkungen:

- Unter z/OS wird MQCNO_CLIENT_BINDING mit MQRC_OPTIONS_ERROR abgelehnt.
- MQCNO_CLIENT_BINDING wird mit MQRC_OPTIONS_ERROR abgelehnt, wenn eine andere MQCNO-Bindungsoption als MQCNO_STANDARD_BINDING angegeben wird.
- MQCNO_CLIENT_BINDING ist für Java oder .NET nicht verfügbar, da sie bei der Auswahl des BIND-Typs über eigene Mechanismen verfügen.
- Ist die Umgebungsvariable MQ_CONNECT_TYPE beim Aufruf von MQCONNX nicht gesetzt, wird versucht, unter Verwendung des Standardbindungstyps eine Standardserverbindung herzustellen. Kann die Serverbibliothek nicht geladen werden, wird versucht, eine Clientverbindung herzustellen.

MQCNO_LOCAL_BINDING

Geben Sie diese Option an, wenn die Anwendung nur eine Serververbindung herstellen soll. Wenn MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING oder MQCNO_SHARED_BINDING ebenfalls angegeben ist, wird stattdessen der jeweilige Verbindungstyp verwendet und in diesem Abschnitt dokumentiert. Andernfalls wird versucht, unter Verwendung des standardmäßigen Bindungstyps eine Standardserververbindung herzustellen. Für MQCNO_LOCAL_BINDING gelten die folgenden Einschränkungen:

- Unter z/OS wird MQCNO_LOCAL_BINDING ignoriert.
- MQCNO_LOCAL_BINDING wird mit MQRC_OPTIONS_ERROR abgelehnt, wenn eine andere MQCNO-Option als MQCNO_RECONNECT_AS_DEF für die Verbindungswiederholung angegeben ist.
- MQCNO_LOCAL_BINDING ist für Java oder .NET nicht verfügbar, da sie bei der Auswahl des BIND-Typs über eigene Mechanismen verfügen.
- Ist die Umgebungsvariable MQ_CONNECT_TYPE beim Aufruf von MQCONNX nicht gesetzt, wird versucht, unter Verwendung des Standardbindungstyps eine Standardserverbindung herzustellen. Kann die Serverbibliothek nicht geladen werden, wird versucht, eine Clientverbindung herzustellen.

Unter AIX, HP-UX, Solaris, Linuxund Windowskönnen Sie die Umgebungsvariable `MQ_CONNECT_TYPE` mit dem im Feld *`Options`* angegebenen Bindungstyp verwenden, um den verwendeten Bindungstyp zu steuern. Wird diese Umgebungsvariablen angegeben, muss sie aufFASTPATH oder STANDARD gesetzt sein; andernfalls wird sie ignoriert. Bei Angabe des Werts für die Umgebungsvariable muss die Groß-/Kleinschreibung berücksichtigt werden (siehe Umgebungsvariable MQCONNX).

Die Umgebungsvariable und das Feld *`Options`* hängen wie folgt zusammen:

- Wenn Sie die Umgebungsvariable übergehen oder einen nicht unterstützten Wert für sie angeben, wird die Verwendung der Fastpath-Bindung ausschließlich durch das Feld *`Options`* bestimmt.
- Wenn Sie der Umgebungsvariablen einen unterstützten Wert geben, wird die Direktaufrufbindung nur verwendet, wenn *sowohl* die Umgebungsvariable als auch das Feld *`Options`* die Direktaufrufbindung angeben.

Optionen für die Verbindungskennung

Diese Optionen werden nur beim Herstellen einer Verbindung zu einem z/OS-Warteschlangenmanager unterstützt; über sie wird die Verwendung der Verbindungskennung *ConnTag* gesteuert. Es kann nur jeweils eine dieser Optionen angegeben werden:

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Diese Option gibt an, dass die Verbindungskennung ausschließlich innerhalb des lokalen Warteschlangenmanagers verwendet werden soll. Wird die Verbindungskennung bereits im lokalen Warteschlangenmanager verwendet, schlägt der MQCONNX-Aufruf mit dem Ursachencode MQRC_CONN_TAG_IN_USE fehl. Die Verwendung der Verbindungskennung an anderer Stelle in der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört, hat keine Auswirkung auf das Ergebnis des Aufrufs.

MQCNO_SERIALIZE_CONN_TAG_QSG

Diese Option gibt an, dass die Verbindungskennung ausschließlich in der Gruppe mit gemeinsamer Warteschlange verwendet werden soll, zu der der lokale Warteschlangenmanager gehört. Wird die Verbindungskennung bereits in der Gruppe mit gemeinsamer Warteschlange verwendet, schlägt der MQCONNX-Aufruf mit dem Ursachencode MQRC_CONN_TAG_IN_USE fehl.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Diese Option gibt an, dass die Verbindungskennung innerhalb des lokalen Warteschlangenmanagers gemeinsam genutzt werden soll. Wird die Verbindungskennung bereits im lokalen Warteschlangenmanager verwendet, kann der MQCONNX-Aufruf erfolgreich ausgeführt werden, wenn die anfordernde Anwendung in demselben Verarbeitungsbereich wie der vorhandene Benutzer der Kennung aktiv ist. Ist dies nicht der Fall, schlägt der MQCONNX-Aufruf mit dem Ursachencode MQRC_CONN_TAG_IN_USE fehl. Die Verwendung der Verbindungskennung an anderer Stelle in der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört, hat keine Auswirkung auf das Ergebnis des Aufrufs.

- Damit die Verbindungskennung gemeinsam genutzt werden kann, müssen die Anwendungen innerhalb desselben MVS-Adressraums aktiv sein. Handelt es sich bei der Anwendung, die die Verbindungskennung verwendet, um eine Clientanwendung, darf MQCNO_RESTRICT_CONN_TAG_Q_MGR nicht angegeben werden.

MQCNO_RESTRICT_CONN_TAG_QSG

Diese Option gibt an, dass die Verbindungskennung innerhalb der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört, gemeinsam genutzt werden soll. Wird die Verbindungskennung bereits in der Gruppe mit gemeinsamer Warteschlange verwendet, kann der MQCONNX-Aufruf ausgeführt werden, wenn die anfordernde Anwendung in demselben Verarbeitungsbereich aktiv und mit demselben Warteschlangenmanager verbunden ist wie der vorhandene Benutzer der Kennung.

Ist dies nicht der Fall, schlägt der MQCONNX-Aufruf mit dem Ursachencode MQRC_CONN_TAG_IN_USE fehl.

- Damit die Verbindungskennung gemeinsam genutzt werden kann, müssen die Anwendungen innerhalb desselben MVS-Adressraums aktiv sein. Handelt es sich bei der Anwendung, die die Verbindungskennung verwendet, um eine Clientanwendung, darf MQCNO_RESTRICT_CONN_TAG_QSG nicht angegeben werden.

Wenn keine dieser Optionen angegeben wird, wird *ConnTag* nicht verwendet. Ist für *Version* eine niedrigere Version als MQCNO_VERSION_3 angegeben, sind diese Optionen ungültig.

Optionen für die gemeinsame Nutzung von Handles

Diese Optionen werden in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux und Windows. Mit ihnen wird die gemeinsame Nutzung von Handles durch unterschiedliche Threads (Einhei-

ten paralleler Verarbeitungsvorgänge) innerhalb desselben Prozesses gesteuert. Es kann nur jeweils eine dieser Optionen angegeben werden:

MQCNO_HANDLE_SHARE_NONE

Diese Option gibt an, dass Verbindungs- und Objekthandles nur von dem Thread verwendet werden können, durch den das Handle zugeordnet wurde (also von dem Thread, von dem der MQCONN-, MQCONNX- oder MQOPEN-Aufruf ausgegeben wurde). Die Handles können nicht von anderen Threads, die zu demselben Prozess gehören, verwendet werden.

MQCNO_HANDLE_SHARE_BLOCK

Diese Option gibt an, dass die Verbindungs- und Objekthandles, die von einem Thread eines Prozesses zugeordnet wurden, auch von anderen Threads verwendet werden können, die zu demselben Prozess gehören. Ein Handle kann jedoch nur jeweils von einem Thread verwendet werden, d. h., es ist nur eine serielle Verwendung von Handles möglich. Versucht ein Thread, ein Handle zu verwendet, das bereits von einem anderen Thread verwendet wird, wird der Aufruf blockiert (d. h., er wartet), bis das Handle wieder zur Verfügung steht.

MQCNO_HANDLE_SHARE_NO_BLOCK

Dieselbe Option wie MQCNO_HANDLE_SHARE_BLOCK, nur wird der Aufruf, wenn das Handle von einem anderen Thread verwendet wird, umgehend mit MQCC_FAILED und MQRC_CALL_IN_PROGRESS beendet; es wird nicht gewartet, bis das Handle wieder zur Verfügung steht.

Ein Thread kann über 0 oder 1 nicht gemeinsam genutztes Handle verfügen:

- Jeder MQCONN- oder MQCONNX-Aufruf, in dem MQCNO_HANDLE_SHARE_NONE angegeben ist, gibt beim ersten Aufruf ein nicht gemeinsam genutztes Handle zurück. Dasselbe nicht gemeinsam genutzte Handle wird auch beim nächsten und bei allen weiteren Aufrufen zurückgegeben (sofern zwischendurch kein MQDISC-Aufruf ausgegeben wird). Für den nächsten Aufruf und alle weiteren Aufrufe wird der Ursachencode MQRC_ALREADY_CONNECTED zurückgegeben.
- Jeder MQCONNX-Aufruf, in dem MQCNO_HANDLE_SHARE_BLOCK oder MQCNO_HANDLE_SHARE_NO_BLOCK angegeben ist, gibt in jedem Aufruf ein neues gemeinsam genutztes Handle zurück.

Objekthandles erben dieselben Eigenschaften für die gemeinsame Nutzung wie das im MQOPEN-Aufruf angegebene Verbindungshandle, von dem das Objekthandle erstellt wurde. Ebenso erben Arbeitseinheiten dieselben Eigenschaften für die gemeinsame Nutzung wie das Verbindungshandle, mit dem die Arbeitseinheit gestartet wurde; wird die Arbeitseinheit in einem Thread unter Verwendung eines gemeinsam genutzten Handles gestartet, kann die Arbeitseinheit in einem anderen Thread unter Verwendung desselben Handles aktualisiert werden.

Wird keine Option für die gemeinsame Nutzung von Handles angegeben, hängt der Standardwert, der übernommen wird, von der jeweiligen Umgebung ab:

- In MTS-Umgebungen (Microsoft Transaction Server) entspricht der Standardwert der Option MQCNO_HANDLE_SHARE_BLOCK.
- In anderen Umgebungen entspricht der Standardwert der Option MQCNO_HANDLE_SHARE_NONE.

Optionen für die Verbindungswiederholung

Über die Optionen für die Verbindungswiederholung wird angegeben, ob die Herstellung einer Verbindung wiederholt werden kann. Nur Clientverbindungen können wiederholt werden.

MQCNO_RECONNECT_AS_DEF

Die Option für die Verbindungswiederholung wird in den Standardwert aufgelöst. Ist kein Standardwert gesetzt, wird der Wert dieser Option in DISABLED aufgelöst. Der Wert dieser Option wird an den Server übergeben und kann über PCF- und MQSC-Befehle abgerufen werden.

MQCNO_RECONNECT

Die Anwendung kann mit jedem Warteschlangenmanager verbunden werden, der mit dem Wert des Parameters QmgrName von MQCONNXkonsistent ist. Die Option MQCNO_RECONNECT sollte nur verwendet werden, wenn zwischen der Clientanwendung und dem Warteschlangenmanager, mit dem ursprünglich eine Verbindung hergestellt wurde, keine Affinität besteht. Der Wert dieser Option wird an den Server übergeben und kann über PCF- und MQSC-Befehle abgerufen werden.

MQCNO_RECONNECT_DISABLED

Die Anwendung kann nicht wieder verbunden werden. Der Wert der Option wird nicht an den Server übergeben.

MQCNO_RECONNECT_Q_MGR

Die Anwendung kann nur mit dem Warteschlangenmanager wieder verbunden werden, mit dem sie ursprünglich verbunden war. Geben Sie diese Option an, wenn ein Client wieder verbunden werden kann, zwischen der Clientanwendung und dem Warteschlangenmanager, mit dem sie ursprünglich verbunden war, jedoch eine Affinität besteht. Geben Sie diesen Wert an, wenn ein Client automatisch wieder mit der Standby-Instanz eines hochverfügbaren Warteschlangenmanagers verbunden werden soll. Der Wert dieser Option wird an den Server übergeben und kann über PCF- und MQSC-Befehle abgerufen werden.

Die Optionen MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED und MQCNO_RECONNECT_Q_MGR sollten nur für Clientanwendungen verwendet werden. Wenn die Optionen für eine Bindungsverbindung verwendet werden, schlägt MQCONNX mit Beendigungscode MQCC_FAILED und Ursachencode MQRC_OPTIONS_ERRORfehl. Automatische Wiederherstellung der Clientverbindung wird von WebSphere MQ -Klassen für Java nicht unterstützt

Optionen für die gemeinsame Nutzung einer Kanalinstanz

Die folgenden Optionen gelten nur für TCP/IP-Clientverbindungen. Für SNA-, SPX- und NetBios-Kanäle werden diese Werte ignoriert und der Kanal ist wie in den vorherigen Produktversionen aktiv.

MQCNO_NO_CONV_SHARING

Bei Angabe dieser Option ist keine gemeinsame Nutzung der Kanalinstanz (Shared Conversations) möglich.

MQCNO_NO_CONV_SHARING kann bei Verbindungen mit einem hohen Datenaufkommen verwendet werden, bei denen es daher auf der Serververbindungsseite der Kanalinstanz, die gemeinsam genutzt wird, zu Konfliktsituationen kommen kann. Das Verhalten bei MQCNO_NO_CNV_SHARING entspricht dem bei sharecnv(1) bei einer Verbindung zu einem Kanal, der die gemeinsame Nutzung einer Kanalinstanz für Verbindungen (Shared Conversations) nicht unterstützt.

MQCNO_ALL_CONVS_SHARE

Diese Option gibt an, dass die gemeinsame Nutzung der Kanalinstanz möglich ist; die Anwendung schränkt die Anzahl der Verbindungen über die Kanalinstanz nicht ein. Diese Option ist der Standardwert.

Wenn die Anwendung angibt, dass die Kanalinstanz gemeinsam genutzt werden kann, die Definition *SharingConversations* (SHARECNV) auf der Serververbindungsseite des Kanals jedoch auf 1 gesetzt ist, erfolgt keine gemeinsame Nutzung und es wird keine Warnung an die Anwendung ausgegeben.

Wenn die Anwendung angibt, dass die gemeinsame Nutzung zulässig ist, die *SharingConversations* -Definition für die Serververbindung jedoch auf null gesetzt ist, wird keine Warnung ausgegeben, und die Anwendung zeigt dasselbe Verhalten wie ein Client in Versionen des Produkts vor Version 7.0. Die Anwendungseinstellung für die gemeinsame Nutzung von Dialogen wird ignoriert.

MQCNO_NO_CONV_SHARING und MQCNO_ALL_CONVS_SHARE schließen sich gegenseitig aus. Werden für eine Verbindung beide Optionen angegeben, wird die Verbindung mit dem Ursachencode MQRC_OPTIONS_ERROR abgelehnt.

Optionen für die Kanaldefinition

Mit den folgenden Optionen wird die Verwendung der Kanaldefinitionsstruktur gesteuert, die in MQCNO übergeben wird:

MQCNO_CD_FOR_OUTPUT_ONLY

Diese Option gibt an, dass die Kanaldefinitionsstruktur MQCNO nur für die Rückgabe des Kanalnamens verwendet werden kann, der in einem erfolgreichen MQCONNX-Aufruf verwendet wurde.

Wird keine gültige Kanaldefinitionsstruktur bereitgestellt, fällt der Aufruf mit dem Ursachencode MQRC_CD_ERROR fehl.

Wenn die Anwendung nicht als Client ausgeführt wird, wird die Option ignoriert.

Der zurückgegebene Kanalname kann in einem anschließenden MQCONNX-Aufruf verwendet werden, in dem mithilfe der Option MQCNO_USE_CD_SELECTION eine erneute Verbindung unter Verwendung derselben Kanaldefinition hergestellt wird. Dies ist hilfreich, wenn die Clientkanaltabelle mehrere zutreffende Kanaldefinitionen enthält.

MQCNO_USE_CD_SELECTION

Diese Option gibt an, dass ein MQCONNX-Aufruf eine Verbindung mithilfe des Kanalnamens herstellen kann, der in der in MQCNO übergebenen Kanaldefinitionsstruktur enthalten ist.

Wurde die Umgebungsvariable MQSERVER gesetzt, wird die in ihr definierte Kanaldefinition verwendet. Wurde MQSERVER nicht gesetzt, wird die Clientkanaltabelle verwendet.

Wird keine Kanaldefinition mit einem entsprechenden Kanalnamen und Warteschlangenmanagernamen gefunden, schlägt der Aufruf mit dem Ursachencode MQRC_Q_MGR_NAME_ERROR fehl.

Wird keine gültige Kanaldefinitionsstruktur bereitgestellt, fällt der Aufruf mit dem Ursachencode MQRC_CD_ERROR fehl.

Wenn die Anwendung nicht als Client ausgeführt wird, wird die Option ignoriert.

Standardoption

Werden keine der oben beschriebenen Optionen benötigt, können Sie die folgende Option verwenden:

MQCNO_NONE

Es werden keine Optionen angegeben.

MQCNO_NONE dient zur Unterstützung der Programmdokumentation. Diese Option ist nicht zur Verwendung mit einer der anderen MQCNO_*-Optionen gedacht; da sie jedoch den Wert 0 hat, kann eine solche Verwendung nicht erkannt werden.

SecurityParmsOffset (MQLONG)
Das Feld "SecurityParmsOffset" gibt (in Bytes) die relative Position der MQCSP-Struktur ab Beginn der MQCNO-Struktur an. Der Offset kann positiv oder negativ sein. Dieses Feld ist ein Eingabefeld, dessen Anfangswert 0 ist.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_5 ist.

Die MQCSP-Struktur ist im Abschnitt „MQCSP - Sicherheitsparameter” auf Seite 315 definiert.

SecurityParmsPtr (PMQCSP)
Das Feld "SecurityParmsPtr" enthält die Adresse der MQCSP-Struktur und wird für die Angabe einer Benutzer-ID und eines Kennworts für die Authentifizierung durch den Berechtigungsservice verwendet. Dieses Feld ist ein Eingabefeld, dessen Anfangswert auf einen Nullzeiger oder auf Nullbytes gesetzt ist.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_5 ist.

Die MQCSP-Struktur ist im Abschnitt „MQCSP - Sicherheitsparameter” auf Seite 315 definiert.

SSLConfigOffset (MQLONG)
Das Feld "SSLConfigOffset" gibt (in Bytes) die relative Position einer MQSCO-Struktur ab Beginn der MQCNO-Struktur an. Der Offset kann positiv oder negativ sein. Dieses Feld ist ein Eingabefeld, dessen Anfangswert 0 ist.

Verwenden Sie das Feld *SSLConfigOffset* nur, wenn die Anwendung, die den MQCONNX-Aufruf ausgibt, als WebSphere MQ-MQI-Client ausgeführt wird. Sie finden Informationen zur Verwendung dieses Felds in der Beschreibung des Felds *SSLConfigPtr*.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_4 ist.

SSLConfigPtr (PMQSCO)
SSLConfigPtr ist ein Eingabefeld. Sein Anfangswert ist bei Programmiersprachen, die Zeiger unterstützen, ein Nullzeiger. Andernfalls ist der Anfangswert eine Bytefolge, die gänzlich aus Nullen besteht.

Verwenden Sie die Felder *SSLConfigPtr* und *SSLConfigOffset* nur, wenn die Anwendung, die den MQCONNX-Aufruf ausgibt, als WebSphere MQ-MQI-Client ausgeführt und als Kanalprotokoll TCP/IP verwendet wird. Wenn die Anwendung nicht als WebSphere MQ-Client ausgeführt oder ein anderes Kanalprotokoll als TCP/IP verwendet wird, werden die Felder *SSLConfigPtr* und *SSLConfigOffset* ignoriert.

Durch die Angabe des Feldes *SSLConfigPtr* oder *SSLConfigOffset* sowie entweder des Feldes *ClientConnPtr* oder *ClientConnOffset* kann die Anwendung die Verwendung von SSL für die Clientverbindung steuern. Werden die SSL-Informationen auf diese Weise angegeben, werden die Umgebungsvariablen MQSSLKEYR und MQSSLCRYP ignoriert. Eventuell vorhandene SSL-bezogene Informationen in der Definitionstabelle für Clientkanäle werden ebenfalls ignoriert.

Die SSL-Informationen können nur in folgenden Aufrufen angegeben werden:

- Im ersten MQCONNX-Aufruf des Clientprozesses oder
- in einem nachfolgenden MQCONNX-Aufruf, wenn alle vorherigen SSL/TLS-Verbindungen mit dem Warteschlangenmanager über den MQDISC-Aufruf aufgehoben wurden.

Die prozessweite SSL-Umgebung kann nur in diesen Zuständen initialisiert werden. Wird bei einer bereits vorhandenen SSL-Umgebung ein MQCONNX-Aufruf ausgegeben, der SSL-Informationen angibt, werden die SSL-Informationen im Aufruf ignoriert und die Verbindung wird unter Verwendung der vorhandenen SSL-Umgebung hergestellt; der Aufruf gibt in diesem Fall den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SSL_ALREADY_INITIALIZED zurück.

Sie können die MQSCO-Struktur auf die gleiche Weise wie die MQCD-Struktur angeben, indem Sie entweder eine Adresse im Feld *SSLConfigPtr* oder eine relative Position im Feld *SSLConfigOffset* angeben. Sie finden Informationen zur entsprechenden Vorgehensweise in der Beschreibung des Feldes *ClientConnPtr*. Allerdings können Sie nur jeweils entweder *SSLConfigPtr* oder *SSLConfigOffset* angeben, da der Aufruf mit dem Ursachencode MQRC_SSL_CONFIG_ERROR fehlschlägt, wenn beide Felder einen Wert ungleich null enthalten.

Sobald der MQCONNX-Aufruf abgeschlossen ist, wird nicht mehr auf die MQSCO-Struktur verwiesen.

Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_4 ist.

Anmerkung: Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

StrucId (MQCHAR4)
StrucId ist immer ein Eingabefeld. Sein Anfangswert lautet MQCNO_STRUC_ID.

Folgende Werte sind möglich:

MQCNO_STRUC_ID
Die ID für die Verbindungsoptionsstruktur.

Für die Programmiersprache C ist auch die Konstante MQCNO_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQCNO_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Version (MQLONG)
Version ist immer ein Eingabefeld. Sein Anfangswert lautet MQCNO_VERSION_1.

Folgende Werte sind möglich:

MQCNO_VERSION_1
Version 1 der Verbindungsoptionsstruktur.

MQCNO_VERSION_2
Version 2 der Verbindungsoptionsstruktur.

MQCNO_VERSION_3
Version 3 der Verbindungsoptionsstruktur.

MQCNO_VERSION_4
Version 4 der Verbindungsoptionsstruktur.

MQCNO_VERSION_5
Verbindungsoptionsstruktur der Version 5.

Diese Version der MQCNO-Struktur ist unter z/OS eine Erweiterung von MQCNO_VERSION_3, während sie auf allen anderen Plattformen eine Erweiterung von MQCNO_VERSION_4 darstellt.

Felder, die nur in den aktuelleren Versionen der Struktur vorliegen, sind in den Beschreibungen der Felder als solche angegeben. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCNO_CURRENT_VERSION
Aktuelle Version der Verbindungsoptionsstruktur.

Anfangswerte und Sprachendeklarationen für MQCNO

Tabelle 488. Anfangswerte der Felder in MQCNO für MQCNO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCNO_STRUC_ID	'CNO¬'
Version	MQCNO_VERSION_1	1
Options	MQCNO_NONE	0
ClientConnOffset	--	0
ClientConnPtr	--	Nullzeiger oder Null Byte
ConnTag	MQCT_NONE	Nullen
SSLConfigPtr	--	Nullzeiger oder Null Byte
SSLConfigOffset	--	0
ConnectionId	--	Nullzeiger oder Null Byte
SecurityParmsOffset	--	Nullzeiger oder Null Byte
SecurityParmsPtr	--	Nullzeiger oder Null Byte

Tabelle 488. Anfangswerte der Felder in MQCNO für MQCNO (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQCNO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQLONG     Options;           /* Options that control the action of
                                   MQCONNX */
  MQLONG     ClientConnOffset;  /* Offset of MQCD structure for client
                                   connection */
  MQPTR      ClientConnPtr;     /* Address of MQCD structure for client
                                   connection */
  MQBYTE128  ConnTag;           /* Queue-manager connection tag */
  PMQSCO     SSLConfigPtr;      /* Address of MQSCO structure for client
                                   connection */
  MQLONG     SSLConfigOffset;   /* Offset of MQSCO structure for client
                                   connection */
  MQBYTE24   ConnectionId;      /* Unique connection identifier */
  MQLONG     SecurityParmsOffset /* Security fields */
  PMQCSP     SecurityParmsPtr   /* Security parameters */
 };
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQCNO based,
  3 StrucId          char(4),       /* Structure identifier */
```

```
  3 Version          fixed bin(31), /* Structure version number */
  3 Options          fixed bin(31), /* Options that control the action
                                       of MQCONNX */
  3 ClientConnOffset fixed bin(31), /* Offset of MQCD structure for
                                       client connection */
  3 ClientConnPtr    pointer,       /* Address of MQCD structure for
                                       client connection */
  3 ConnTag          char(128),     /* Queue-manager connection tag */
  3 SSLConfigPtr     pointer,       /* Address of MQSCO structure for
                                       client connection */
  3 SSLConfigOffset  fixed bin(31), /* Offset of MQSCO structure for
                                       client connection */
  3 ConnectionId     char(24),      /* Unique connection identifier
  3 SecurityParmsOffset fixed bin(31); /* Offset of MQCSP structure for
                                       security parameters */
  3 SecurityParmsPtr pointer,       /* Address of MQCSP structure for
                                       security parameters */
```

Deklaration in High Level Assembler

```
MQCNO                   DSECT
MQCNO_STRUCID           DS   CL4    Structure identifier
MQCNO_VERSION           DS   F      Structure version number
MQCNO_OPTIONS           DS   F      Options that control the action of
*                                   MQCONNX
MQCNO_CLIENTCONNOFFSET  DS   F      Offset of MQCD structure for client
*                                   connection
MQCNO_CLIENTCONNPTR     DS   F      Address of MQCD structure for client
*                                   connection
MQCNO_CONNTAG           DS   XL128  Queue-manager connection tag
*
MQCNO_CONNECTIONID      DS   XL24   Unique connection identifier
*
MQCNO_SSLCONFIGOFFSET   DS   F      Offset of MQCSP structure for security
*                                   parameters
MQCNO_SSLCONFIGPTR      DS   F      Address of MQCSP structure for security
*                                   parameters
MQCNO_LENGTH            EQU  *-MQCNO
                        ORG  MQCNO
MQCNO_AREA              DS   CL(MQCNO_LENGTH)
```

Deklaration in Visual Basic

```
Type MQCNO
  StrucId          As String*4  'Structure identifier'
  Version          As Long      'Structure version number'
  Options          As Long      'Options that control the action of'
                                'MQCONNX'
  ClientConnOffset As Long      'Offset of MQCD structure for client'
                                'connection'
  ClientConnPtr    As MQPTR     'Address of MQCD structure for client'
                                'connection'
  ConnTag          As MQBYTE128 'Queue-manager connection tag'
  SSLConfigPtr     As MQPTR     'Address of MQSCO structure for client'
                                'connection'
  SSLConfigOffset  As Long      'Offset of MQSCO structure for client'
                                'connection'
  ConnectionId     As MQBYTE24  'Unique connection identifier'
  SecurityParmsOffset As Long   'Offset of MQCSP structure for security'
                                'parameters'
  SecurityParmsPtr As MQPTR     'Address of MQCSP structure for security'
                                'parameters'
  End Type
```

MQCSP - Sicherheitsparameter

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 489. Felder in MQCSP		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
AuthenticationType	Authentifizierungstyp	AuthenticationType
Reserved1	Für die Zeigerausrichtung unter IBM i erforderlich	Reserved1
CSPUserIdPtr	Adresse der Benutzer-ID	CSPUserIdPtr
CSPUserIdOffset	Relative Position der Benutzer-ID	CSPUserIdOffset
CSPUserIdLength	Länge der Benutzer-ID	CSPUserIdLength
Reserved2	Für die Zeigerausrichtung unter IBM i erforderlich	Reserved2
CSPPasswordPtr	Adresse des Kennworts	CSPPasswordPtr
CSPPasswordOffset	Relative Position des Kennworts	CSPPasswordOffset
CSPPasswordLength	Länge des Kennworts	CSPPasswordLength

Übersicht über MQCSP

Verfügbarkeit: Alle WebSphere MQ-Produkte.

Zweck: Die MQCSP-Struktur ermöglicht dem Berechtigungsservice die Authentifizierung einer Benutzer-ID und eines Kennworts. Sie geben die Sicherheitsparameterstruktur der MQCSP-Verbindung in einem MQCONNX-Aufruf an.

Zeichensatz und Codierung: Die Daten in MQCSP müssen den Zeichensatz und die Codierung des lokalen Warteschlangenmanagers aufweisen. Diese werden durch das Warteschlangenmanagerattribut *CodedCharSetId* bzw. durch MQENC_NATIVE angegeben.

Felder für MQCSP

Die MQCSP-Struktur enthält die folgenden Felder; die Felder werden in **alphabetischer Reihenfolge** beschrieben:

AuthenticationType (MQLONG)
AuthenticationType ist ein Eingabefeld. Sein Anfangswert lautet MQCSP_AUTH_NONE.

Dies ist der Typ der durchzuführenden Authentifizierung. Gültige Werte sind:

MQCSP_AUTH_NONE
Verwenden Sie keine Benutzer-ID- und Kennwortfelder.

MQCSP_AUTH_USER_ID_AND_PWD
Die Felder für Benutzer-ID und Kennwort werden authentifiziert.

CSPPasswordLength (MQLONG)
Dieses Feld ist die Länge des Kennworts, das in der Authentifizierung verwendet werden soll.

Die maximale Länge des Kennworts hängt von der jeweiligen Plattform ab; der Abschnitt Benutzer-IDs enthält weitere Informationen hierzu. Übersteigt die Kennwortlänge die maximal zulässige Länge, schlägt die Authentifizierungsanforderung mit MQRC_NOT_AUTHORIZED fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

CSPPasswordOffset (MQLONG)
Dieses Feld enthält (in Bytes) die relative Position des in der Authentifizierung zu verwendenden Kennworts. Der Offset kann positiv oder negativ sein.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

CSPPasswordPtr (MQPTR)
Dies ist die Adresse in Byte des Kennworts, das in der Authentifizierung verwendet werden soll.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_5 ist.

CSPUserIdLength (MQLONG)
Dieses Feld ist die Länge der Benutzer-ID, die in der Authentifizierung verwendet werden soll.

Die maximale Länge der Benutzer-ID hängt von der jeweiligen Plattform ab; der Abschnitt Benutzer-IDs enthält weitere Informationen hierzu. Übersteigt die Länge der Benutzer-ID die maximal zulässige Länge, schlägt die Authentifizierungsanforderung mit MQRC_NOT_AUTHORIZED fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

CSPUserIdOffset (MQLONG)
Dieses Feld enthält (in Bytes) die relative Position der in der Authentifizierung zu verwendenden Benutzer-ID. Der Offset kann positiv oder negativ sein.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

CSPUserIdPtr (MQPTR)
Dies ist die Adresse in Byte der Benutzer-ID, die in der Authentifizierung verwendet werden soll.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQCNO_VERSION_5 ist.

Reserved1 (MQBYTE4)
Ein reserviertes Feld, das für die Zeigerausrichtung unter IBM i erforderlich ist.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds sind alles Nullen.

Reserved2 (MQBYTE8)
Ein reserviertes Feld, das für die Zeigerausrichtung unter IBM i erforderlich ist.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds sind alles Nullen.

StrucId (MQCHAR4)
Struktur-ID.

Folgende Werte sind möglich:

MQCSP_STRUC_ID
Die ID für die Sicherheitsparameterstruktur.

Für die Programmiersprache C ist auch die Konstante MQCSP_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQCSP_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds lautet MQCSPSTRUC_ID.

Version (MQLONG)
Strukturversionsnummer.

Folgende Werte sind möglich:

MQCSP_VERSION_1
Version 1 der Sicherheitsparameterstruktur.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCSP_CURRENT_VERSION
Aktuelle Version der Sicherheitsparameterstruktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds lautet MQCSP_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQCSP

Tabelle 490. Anfangswerte der Felder in MQCSP für MQCSP

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCSP_STRUC_ID	'CSP¬'
Version	MQCSP_VERSION_1	1
AuthenticationType	--	MQCSP_AUTH_NONE
Reserved1	--	Nullzeichenfolge oder Leerzeichen.
CSPUserIdPtr	--	Nullzeiger oder Null Byte
CSPUserIdOffset	--	0
CSPUserIdLength	--	0
Reserved2	--	Nullzeichenfolge oder Leerzeichen.
CSPPasswordPtr	--	Nullzeiger oder Null Byte
CSPPasswordOffset	--	0
CSPPasswordLength	--	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQCSP_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer
                                    alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer
                                    alignment */
                                 alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
```

```
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
  };
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQCSP based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 AuthenticationType fixed bin(31),  /* Type of authentication */
  3 Reserved1         char(4),     /* Required for IBM i pointer
                                      alignment */
  3 CSPUserIdPtr      pointer,      /* Address of user ID */
  3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
  3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
  3 Reserved2         char(8),     /* Required for IBM i pointer
                                      alignment */
  3 CSPPasswordPtr    pointer,      /* Address of password */
  3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
  3 CSPPasswordLength fixed bin(31); /* Length of user ID */
```

Deklaration in Visual Basic

```
Type MQCSP
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  AuthenticationType As Long      'Type of authentication'
  Reserved1         As MQBYTE4    'Required for IBM i pointer'
                                  'alignment'
  CSPUserIdPtr      As MQPTR      'Address of user ID'
  CSPUserIdOffset   As Long       'Offset of user ID'
  CSPUserIdLength   As Long       'Length of user ID'
  Reserved2         As MQBYTE8    'Required for IBM i pointer'
                                  'alignment'
  CSPPasswordPtr    As MQPTR      'Address of password'
  CSPPasswordOffset As Long       'Offset of password'
  CSPPasswordLength As Long       'Length of password'
End Type
```

MQCTLO – Struktur der Optionen für die Callback-Steuerung

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. Struktur, die die Callback-Funktion für die Steuerung angibt

Tabelle 491. Felder in MQCTLO		
Feld	**Beschreibung**	**Thema**
StrucID	Struktur-ID	StrucID
Version	Strukturversionsnummer	Version
Options	Optionen	Options
Reserved	Reserviertes Feld	Options
ConnectionArea	Feld für zu verwendende Callback-Funktion	ConnectionArea

Überblick für MQCTLO

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSund WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind. Überblick über die MQCTLO-Struktur.

Zweck: Mithilfe der MQCTLO-Struktur werden Optionen für eine Callback-Steuerungsfunktion angegeben.

Die Struktur ist ein Eingabe-und Ausgabeparameter im Aufruf „MQCTL - Callbacks steuern” auf Seite 672 .

Version: Die aktuelle Version von MQCTLO ist MQCTLO_VERSION_1.

Zeichensatz und Codierung: Die Daten in MQCTLO müssen im Zeichensatz im Attribut *CodedCharSetId* des Warteschlangenmanagers enthalten sein, die Codierung des lokalen Warteschlangenmanagers wird über MQENC_NATIVE angegeben. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQCTLO

Alphabetische Liste der Felder für die MQCTLO-Struktur.

Die MQCTLO-Struktur enthält folgende Felder (die Felder werden in alphabetischer Reihenfolge beschrieben):

ConnectionArea (MQPTR)
Struktur für Steuerungsoptionen - Feld ConnectionArea

Dies ist ein Feld, das von einer Callback-Funktion verwendet werden kann.

Der Warteschlangenmanager trifft keine Entscheidungen auf der Basis des Inhalts dieses Felds und er wird unverändert an das Feld „ConnectionArea (MQPTR)” auf Seite 266 in der MQCBC-Struktur übergeben, bei dem es sich um einen Eingabeparameter für den Callback handelt.

Dieses Feld wird bei allen Operationen außer MQOP_START und MQOP_START_WAIT ignoriert.

Dies ist ein Ein-/Ausgabefeld für die Callback-Funktion. Der Anfangswert dieses Felds ist ein Nullzeiger oder null Bytes.

Optionen (MQLONG)
Struktur für Steuerungsoptionen - Feld Options

Optionen zur Steuerung der Aktion von MQCTL.

MQCTLO_FAIL_IF_QUIESCING
Erzwingt ein Fehlschlagen des MQCTL-Aufrufs, wenn sich der Warteschlangenmanager oder die Verbindung im Quiescestatus befindet.

Geben Sie MQGMO_FAIL_IF_QUIESCING in den an den MQCB-Aufruf übergebenen MQGMO-Optionen an, um einen Hinweis an die Nachrichtenkonsumenten auszulösen, wenn sie in den Quiescemodus versetzt werden.

MQCTLO_THREAD_AFFINITY
Diese Option informiert das System, dass die Anwendung voraussetzt, dass alle Nachrichtenkonsumenten für dieselbe Verbindung in demselben Thread aufgerufen werden. Dieser Thread wird für alle Aufrufe der Konsumenten verwendet, bis die Verbindung gestoppt wird.

Standardoption: Falls Sie keine der beschriebenen Optionen benötigen, verwenden Sie folgende Option:

MQCTLO_NONE
Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an. MQCTLO_NONE dient zur Unterstützung der Programmdokumentation und sollte nicht mit anderen Optionen verwendet werden; da sie jedoch den Wert null hat, kann ihre Verwendung nicht erkannt werden.

Dies ist ein Eingabefeld. Der Anfangswert des Felds *Options* ist MQCTLO_NONE.

Reserved (MQLONG)

Dies ist ein reserviertes Feld. Der Wert muss null sein.

StrucId (MQCHAR4)
Struktur für Steuerungsoptionen - Feld StrucId

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQCTLO_STRUC_ID
ID der Struktur für die Steueroptionen.

Für die Progarmmiersprache C ist auch die Konstante MQCTLO_STRUC_ID_ARRAY definiert; diese hat denselben Wert wie MQCTLO_STRUC_ID, aber es handelt sich dabei nicht um eine Zeichenfolge, sondern um eine Gruppe von Zeichen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCTLO_STRUC_ID.

Version (MQLONG)
Struktur für Steuerungsoptionen - Feld Version

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQCTLO_VERSION_1
Steuerungsoptionsstruktur Version-1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCTLO_CURRENT_VERSION
Aktuelle Version der Struktur der Steueroptionen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQCTLO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQCTLO
Struktur für Steuerungsoptionen - Anfangswerte

Tabelle 492. Anfangswerte der Felder in MQCTLO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQCTLO_STRUC_ID	'CTLO'
Version	MQCTLO_VERSION_1	1
Options	MQCTLO_NONE	Nullen
Reserved	Reserviertes Feld	
ConnectionArea	--	Nullzeiger oder Null Byte

Tabelle 492. Anfangswerte der Felder in MQCTLO (Forts.)		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQCTLO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQCTLO MyCTLO = {MQCTLO_DEFAULT};
```

Deklaration in Programmiersprache C
Struktur für Steuerungsoptionen - Deklaration für Programmiersprache C

```
typedef struct tagMQCTLO MQCTLO;
 struct tagMQCTLO {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    Options;        /* Options that control the action of MQCTL */
  MQLONG    Reserved;       /* Reserved field */


MQPTR     ConnectionArea; /* Connection work area passed to the function */
 };
```

COBOL-Delaration

```
** 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
```

Deklaration in PL/I

```
dcl
 1 MQCTLO based,
   3 StrucId             char(4),       /* Structure identifier */
   3 Version             fixed bin(31), /* Structure version */
   3 Options             fixed bin(31), /* Options */
   3 Reserved            fixed bin(31),
   3 ConnectionArea      pointer;       /* Connection work area */
```

MQDH - Verteilerheader

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 493. Felder in MQDH		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version

Tabelle 493. Felder in MQDH (Forts.)		
Feld	**Beschreibung**	**Thema**
StrucLength	Länge der MQDH-Struktur plus der folgenden Datensätze	StrucLength
Encoding	Numerische Codierung der Daten, die auf eine Feldgruppe mit MQPMR-Datensätzen folgen	Encoding
CodedCharSetId	Zeichensatzkennung der Daten, die auf eine Feldgruppe mit MQPMR-Datensätzen folgen	CodedCharSetId
Format	Formatname der Daten, die auf eine Feldgruppe mit MQPMR-Datensätzen folgen	Format
Flags	Allgemeine Flags	Flags
PutMsgRecFields	Flags, die anzeigen, welche MQPMR-Felder vorhanden sind	PutMsgRecFields
RecsPresent	Anzahl der vorhandenen Objektdatensätze	RecsPresent
ObjectRecOffset	Relative Position des ersten Objektdatensatzes ab Beginn der MQDH-Struktur	ObjectRecOffset
PutMsgRecOffset	Relative Position des ersten Nachrichteneinreihungssatzes ab Beginn der MQDH-Struktur	PutMsgRecOffset

Übersicht über MQDH

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQDH-Struktur beschreibt die zusätzlichen Daten, die in einer Nachricht enthalten sind, wenn diese Nachricht als Verteilerlistennachricht in einer Übertragungswarteschlange gespeichert ist. Eine Verteilerlistennachricht ist eine Nachricht, die an mehrere Zielwarteschlangen gesendet wird. Die zusätzlichen Daten bestehen aus der MQDH-Struktur, auf die eine Feldgruppe mit MQOR-Datensätzen und eine Feldgruppe mit MQPMR-Datensätzen folgen.

Diese Struktur wird von Fachanwendungen verwendet, die Nachrichten direkt in Übertragungswarteschlangen einreihen oder Nachrichten aus Übertragungswarteschlangen entfernen (beispielsweise Nachrichtenkanalagenten).

Anwendungen, die Nachrichten in Verteilerlisten einreihen möchten, dürfen diese Struktur nicht verwenden. Stattdessen müssen sie mit der MQOD-Struktur die Zieladressen in der Verteilerliste definieren und die MQPMO-Struktur für die Angabe von Nachrichteneigenschaften oder den Empfang von Informationen zu den Nachrichten verwenden, die an die einzelnen Zieladressen gesendet werden.

Formatname: MQFMT_DIST_HEADER.

Zeichensatz und Codierung: Die Daten in MQDH müssen den Zeichensatz haben, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und die Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE festgelegt wird.

Legen Sie an folgenden Stellen den Zeichensatz und die Codierung der MQDH-Struktur in den Feldern *CodedCharSetId* und *Encoding* fest:

- im MQMD (wenn sich die MQDH-Struktur am Anfang der Nachrichtendaten befindet) oder
- in der Headerstruktur, die der MQDH-Struktur vorangeht (alle anderen Fälle).

Verwendung: Wenn eine Anwendung eine Nachricht in eine Verteilerliste einreiht und einige oder alle Ziele ferne Ziele sind, stellt der Warteschlangenmanager den Anwendungsnachrichtendaten die MQXQH- und MQDH-Struktur voran und fügt die Nachricht der entsprechenden Übertragungswarteschlange hinzu.

Die Daten treten daher in der folgenden Reihenfolge auf, wenn sich die Nachricht in einer Übertragungswarteschlange befindet:

- MQXQH-Struktur
- MQDH-Struktur plus Feldgruppen mit MQOR- und MQPMR-Datensätzen
- Anwendungsnachrichtendaten

Der Warteschlangenmanager kann abhängig von den Zieladressen mehrere dieser Nachrichten generieren und diese in verschiedene Übertragungswarteschlangen stellen. In diesem Fall geben die MQDH-Strukturen in den Nachrichten unterschiedliche Untergruppen der Ziele an, die durch die von der Anwendung geöffnete Verteilerliste definiert werden.

Eine Anwendung, die eine Verteilerlistennachricht direkt in eine Übertragungswarteschlange einreiht, muss die oben beschriebene Reihenfolge einhalten und sicherstellen, dass die MQDH-Struktur korrekt ist. Falls die MQDH-Struktur nicht gültig ist, kann der Warteschlangenmanager den MQPUT- oder MQPUT1-Aufruf mit dem Ursachencode MQRC_DH_ERROR unterlassen.

Sie können Nachrichten in Form einer Verteilerliste nur dann in einer Warteschlange speichern, wenn Sie die Unterstützung von Verteilerlistennachrichten in der Definition der Warteschlange festgelegt haben (weitere Informationen finden Sie unter dem Warteschlangenattribut *DistLists*, das im Abschnitt „Attribute für Warteschlangen” auf Seite 833 beschrieben wird). Reiht eine Anwendung eine Verteilerlistennachricht direkt in eine Warteschlange ein, die keine Verteilerlisten unterstützt, teilt der Warteschlangenmanager die Verteilerlistennachricht in einzelne Nachrichten auf und stellt stattdessen diese in die Warteschlange.

Felder für MQDH

Die MQDH-Struktur enthält die folgenden Felder; die Felder werden in **Alphabetische Reihenfolge**beschrieben:

CodedCharSetId (MQLONG)

Dieses Feld enthält die Zeichensatzkennung der Daten, die auf die Feldgruppen mit MQOR- und MQPMR-Datensätzen folgen; es wird nicht für die Zeichendaten in der MQDH-Struktur selbst verwendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Sie können den folgenden Spezialwert verwenden:

MQCCSI_INHERIT
Zeichensatz-ID dieser Struktur übernehmen.

Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Sofern kein Fehler auftritt, gibt der MQGET-Aufruf nicht den Wert MQCCSI_INHERIT zurück.

Sie können MQCCSI_INHERIT nicht verwenden, wenn das Feld *PutApplType* im MQMD den Wert MQAT_BROKER enthält.

Dieser Wert wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowsund WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

Encoding (MQLONG)

Dieses Feld enthält die numerische Codierung der Daten, die auf die Feldgruppen mit MQOR- und MQPMR-Datensätzen folgen; es wird nicht für die numerischen Daten in der MQDH-Struktur selbst verwendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Feldes ist 0.

Flags (MQLONG)

Sie können das folgende Flag angeben:

MQDHF_NEW_MSG_IDS
Für jede Zieladresse in der Verteilerliste wird eine neue Nachrichten-ID generiert. Legen Sie dieses Flag nur fest, wenn keine Nachrichteneinreihungssätze vorhanden sind oder wenn die Datensätze zwar vorhanden sind, das Feld *MsgId* jedoch nicht enthalten.

Dieses Flag verzögert die Generierung der Nachrichten-IDs bis zur endgültigen Aufteilung der Verteilerlistennachricht in einzelne Nachrichten. Dadurch wird die Menge der Steuerinformationen minimiert, die mit der Verteilerlistennachricht übertragen werden müssen.

Wenn eine Anwendung eine Nachricht in eine Verteilerliste stellt, legt der Warteschlangenmanager das Flag MQDHF_NEW_MSG_IDS in der von ihm generierten MQDH-Struktur fest, sofern die folgenden beiden Bedingungen erfüllt sind:

- Von der Anwendung werden keine Nachrichteneinreihungssätze bereitgestellt oder das Feld *MsgId* ist nicht in den bereitgestellten Datensätzen enthalten.
- Das Feld *MsgId* im MQMD ist auf MQMI_NONE gesetzt oder das Feld *Options* in der MQPMO-Struktur enthält das Flag MQPMO_NEW_MSG_ID.

Falls keine Flags erforderlich sind, geben Sie Folgendes an:

MQDHF_NONE
Es wurden keine Flags angegeben. MQDHF_NONE wird zur Unterstützung der Programmdokumentation definiert. Diese Konstante ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

Der Anfangswert dieses Felds lautet MQDHF_NONE.

Format (MQCHAR8)

Dieses Feld enthält den Formatnamen der Daten, die auf die Feldgruppen mit MQOD- und MQPMR-Datensätzen folgen (je nachdem, welche zuletzt auftreten).

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds entsprechen denen für das Feld *Format* in MQMD.

Der Anfangswert dieses Felds ist MQFMT_NONE.

ObjectRecOffset (MQLONG)

Dieses Feld gibt die relative Position (in Bytes) des ersten Datensatzes in der Feldgruppe mit MQOR-Objektdatensätzen an, die die Namen der Zielwarteschlangen enthalten. Diese Feldgruppe enthält die unter *RecsPresent* angegebene Anzahl an Datensätzen. Diese Datensätze (plus aller Bytes, die zwischen dem ersten Objektdatensatz und dem vorherigen Feld übersprungen werden) zählen zu der Länge, die durch das Feld *StrucLength* angegeben wird.

Da eine Verteilerliste immer mindestens eine Zieladresse enthalten muss, muss das Feld *ObjectRecOffset* immer einen größeren Wert als null haben.

Der Anfangswert dieses Feldes ist 0.

PutMsgRecFields (MQLONG)

Sie können keines oder aber auch mehrere der folgenden Flags angeben:

MQPMRF_MSG_ID
Nachrichten-ID-Feld ist vorhanden.

MQPMRF_CORREL_ID
Korrelations-ID-Feld ist vorhanden.

MQPMRF_GROUP_ID
Gruppen-ID-Feld ist vorhanden.

MQPMRF_FEEDBACK
Feedbackfeld ist vorhanden.

MQPMRF_ACCOUNTING_TOKEN
Feld für das Abrechnungstoken vorhanden.

Falls keine MQPMR-Felder vorhanden sind, geben Sie Folgendes an:

MQPMRF_NONE
Es sind keine Felder mit Nachrichteneinreihungssätzen vorhanden. MQPMRF_NONE dient zur Unterstützung der Programmdokumentation. Diese Konstante ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

Der Anfangswert dieses Felds ist MQPMRF_NONE.

PutMsgRecOffset (MQLONG)

Dieses Feld gibt die relative Position (in Bytes) des ersten Datensatzes in der Feldgruppe mit MQPMR-Nachrichteneinreihungssätzen an, die die Nachrichteneigenschaften enthalten. Falls vorhanden, enthält diese Feldgruppe die unter *RecsPresent* angegebene Anzahl an Datensätzen. Diese Datensätze (plus aller Bytes, die zwischen dem ersten Nachrichteneinreihungssatz und dem vorherigen Feld übersprungen werden) zählen zu der Länge, die durch das Feld *StrucLength* angegeben wird.

Nachrichteneinreihungssätze sind optional; werden keine Datensätze bereitgestellt, hat das Feld *PutMsgRecOffset* den Wert null und das Feld *PutMsgRecFields* enthält den Wert MQPMRF_NONE.

Der Anfangswert dieses Feldes ist 0.

RecsPresent (MQLONG)

Dieses Feld enthält die Anzahl der Zieladressen. Da eine Verteilerliste immer mindestens eine Zieladresse enthalten muss, muss das Feld *RecsPresent* immer einen größeren Wert als null haben.

Der Anfangswert dieses Feldes ist 0.

StrucId (MQCHAR4)

Folgende Werte sind möglich:

MQDH_STRUC_ID
Die ID für die Struktur des Verteilungsheaders.

Für die Programmiersprache C ist auch die Konstante MQDH_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQDH_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Der Anfangswert dieses Felds lautet MQDH_STRUC_ID.

StrucLength (MQLONG)

Dieses Feld gibt die Anzahl Byte ab dem Anfang der MQDH-Struktur bis zum Anfang der Nachrichtendaten gefolgt von Arrays mit MQOR- und MQPMR-Datensätzen an. Die Daten treten in der folgenden Reihenfolge auf:

- MQDH, Struktur
- Feldgruppe mit MQOR-Datensätzen
- Feldgruppe mit MQPMR-Datensätzen
- Nachrichtendaten

Auf die Arrays der MQOR- und MQPMR-Datensätze wird durch den Offset in der MQDH-Struktur verwiesen. Wenn diese relativen Positionen zu nicht belegten Bytes zwischen einer oder mehreren Instanzen der MQDH-Struktur, der Datensatzfeldgruppen und der Nachrichtendaten führen, müssen diese nicht belegten Bytes zwar in den Wert von *StrucLength* einbezogen werden, der Inhalt dieser Bytes wird

jedoch nicht vom Warteschlangenmanager beibehalten. Dabei hat das Array von MQPMR-Datensätzen Vorrang vor dem Array von MQOR-Datensätzen.

Der Anfangswert dieses Feldes ist 0.

Version (MQLONG)

Folgende Werte sind möglich:

MQDH_VERSION_1
Die Versionsnummer der Struktur des Verteilungsheaders.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQDH_CURRENT_VERSION
Aktuelle Version der Verteilungsheaderstruktur

Der Anfangswert dieses Felds lautet MQDH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQDH

Tabelle 494. Anfangswerte der Felder in MQDH für MQDH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQDH_STRUC_ID	'DH¬¬'
Version	MQDH_VERSION_1	1
StrucLength	--	0
Encoding	--	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQDHF_NONE	0
PutMsgRecFields	MQPMRF_NONE	0
RecsPresent	--	0
ObjectRecOffset	--	0
PutMsgRecOffset	--	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQDH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQDH MyDH = {MQDH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQDH MQDH;
struct tagMQDH {
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Length of MQDH structure plus following
                                MQOR and MQPMR records */
  MQLONG   Encoding;         /* Numeric encoding of data that follows
                                the MQOR and MQPMR records */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows the MQOR and MQPMR records */
  MQCHAR8  Format;           /* Format name of data that follows the
```

```
                                MQOR and MQPMR records */
  MQLONG   Flags;            /* General flags */
  MQLONG   PutMsgRecFields;  /* Flags indicating which MQPMR fields are
                                present */
  MQLONG   RecsPresent;      /* Number of MQOR records present */
  MQLONG   ObjectRecOffset;  /* Offset of first MQOR record from start
                                of MQDH */
  MQLONG   PutMsgRecOffset;  /* Offset of first MQPMR record from start
                                of MQDH */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQDH based,
   3 StrucId         char(4),       /* Structure identifier */
   3 Version         fixed bin(31), /* Structure version number */
   3 StrucLength     fixed bin(31), /* Length of MQDH structure plus
                                       following MQOR and MQPMR
                                       records */
   3 Encoding        fixed bin(31), /* Numeric encoding of data that
                                       follows the MQOR and MQPMR
                                       records */
   3 CodedCharSetId  fixed bin(31), /* Character set identifier of data
                                       that follows the MQOR and MQPMR
                                       records */
   3 Format          char(8),       /* Format name of data that follows
                                       the MQOR and MQPMR records */
   3 Flags           fixed bin(31), /* General flags */
   3 PutMsgRecFields fixed bin(31), /* Flags indicating which MQPMR
                                       fields are present */
   3 RecsPresent     fixed bin(31), /* Number of MQOR records present */
   3 ObjectRecOffset fixed bin(31), /* Offset of first MQOR record from
                                       start of MQDH */
   3 PutMsgRecOffset fixed bin(31); /* Offset of first MQPMR record from
                                       start of MQDH */
```

Deklaration in Visual Basic

```
Type MQDH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
  StrucLength     As Long     'Length of MQDH structure plus following'
                              'MQOR and MQPMR records'
  Encoding        As Long     'Numeric encoding of data that follows'
```

```
                              'the MQOR and MQPMR records'
   CodedCharSetId  As Long     'Character set identifier of data that'
                              'follows the MQOR and MQPMR records'
   Format          As String*8 'Format name of data that follows the'
                              'MQOR and MQPMR records'
   Flags           As Long     'General flags'
   PutMsgRecFields As Long     'Flags indicating which MQPMR fields are'
                              'present'
   RecsPresent     As Long     'Number of MQOR records present'
   ObjectRecOffset As Long     'Offset of first MQOR record from start'
                              'of MQDH'
   PutMsgRecOffset As Long     'Offset of first MQPMR record from start'
                              'of MQDH'
End Type
```

MQDLH - Header für nicht zustellbare Nachrichten

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 495. Felder in MQDLH

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Reason	Ursache für den Eingang der Nachricht in der Warteschlange für nicht zustellbare Nachrichten	Reason
DestQName	Name der ursprünglichen Zielwarteschlange	DestQName
DestQMgrName	Name des ursprünglichen Zielwarteschlangenmanagers	DestQMgrName
Encoding	Numerische Codierung der Daten, die auf die MQDLH-Struktur folgen	Encoding
CodedCharSetId	Zeichensatzkennung der Daten, die auf die MQDLH-Struktur folgen	CodedCharSetId
Format	Formatname der Daten, die auf die MQDLH-Struktur folgen	Format
PutApplType	Typ der Anwendung, von der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde	PutApplType
PutApplName	Name der Anwendung, von der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde	PutApplName
PutDate	Datum, an dem die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde	PutDate
PutTime	Uhrzeit, zu der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde	PutTime

Übersicht über MQDLH

Verfügbarkeit: Alle WebSphere MQ-Plattformen.

Zweck: Die MQDLH-Struktur beschreibt die Informationen, die als Präfix für die Anwendungsnachrichtendaten von Nachrichten in der Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) verwendet werden. Eine Nachricht kann in der Warteschlange für nicht zustellbare Nachrichten

eintreffen, weil sie vom Warteschlangenmanager oder Nachrichtenkanalagenten an diese Warteschlange umgeleitet wurde, oder weil sie von einer Anwendung direkt in diese Warteschlange eingereiht wurde.

Formatname: MQFMT_DEAD_LETTER_HEADER.

Zeichensatz und Codierung: Für die Felder in der MQDLH-Struktur gelten der Zeichensatz und die Codierung, die durch die Felder *CodedCharSetId* und *Encoding* angegeben werden. Diese werden in der Headerstruktur angegeben, die MQDLH vorangeht, oder sich in der MQMD-Struktur befindet, falls sich die MQDLH-Struktur am Anfang der Anwendungsnachrichtendaten befindet.

Um in Warteschlangennamen gültige Zeichen bereitstellen zu können, muss der Zeichensatz Einzelbytezeichen unterstützen.

Wenn die WMQ-Klassen für Java/JMS verwendet werden und die in der MQMD-Struktur definierte Codepage von der JVM (Java Virtual Machine) unterstützt wird, wird der UTF-8-Zeichensatz für die MQDLH-Struktur verwendet.

Verwendung: Anwendungen, die Nachrichten direkt in die Warteschlange für nicht zustellbare Nachrichten einreihen, müssen für die Nachrichtendaten eine MQDLH-Struktur als Präfix verwenden und die Felder mit entsprechenden Werten initialisieren. Der Warteschlangenmanager verlangt jedoch nicht, dass eine MQDLH-Struktur vorhanden ist oder dass gültige Werte für die Felder angegeben wurden.

Wenn eine Nachricht zu lang ist, um sie in die Warteschlange für nicht zustellbare Nachrichten einreihen zu können, muss die Anwendung eine der folgenden Maßnahmen ergreifen:

- Abschneiden der Nachrichtendaten auf eine passende Länge für die Warteschlange für nicht zustellbare Nachrichten.
- Aufzeichnen der Nachricht im Zusatzspeicher und Einfügen einer Ausnahmeberichtsnachricht mit einem entsprechenden Hinweis in die Warteschlange für nicht zustellbare Nachrichten.
- Nachricht verwerfen und an den Sender einen Fehler zurückgeben. Wenn die Nachricht eine (möglicherweise) kritische Nachricht ist, darf diese Maßnahme nur ergriffen werden, wenn der Absender bekanntermaßen über eine Nachrichtenkopie verfügt. Ein Beispiel hierfür wäre eine Nachricht, die von einem Nachrichtenkanalagenten über einen Kommunikationskanal empfangen wird.

Welche der obigen Maßnahmen geeignet ist (sofern überhaupt eine geeignet ist), hängt von der jeweiligen Konstruktion der Anwendung ab.

Der Warteschlangenmanager führt eine besondere Verarbeitung durch, wenn eine Nachricht, die ein Segment ist, mit einer vorangestellten MQDLH-Struktur eingereiht wird; nähere Informationen hierzu finden Sie in der Beschreibung der MQMDE-Struktur.

Einreihen von Nachrichten in die Warteschlange für nicht zustellbare Nachrichten: Wird eine Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht, muss die MQMD-Struktur, die für den MQPUT- oder MQPUT1-Aufruf verwendet wird, mit dem MQMD identisch sein, der der Nachricht zugeordnet ist (in der Regel ist dies der MQMD, der vom MQGET-Aufruf zurückgegeben wird). Hierbei gelten jedoch folgende Ausnahmen:

- Setzen Sie die Felder *CodedCharSetId* und *Encoding* auf den Zeichensatz und die Codierung, die für die Felder in der MQDLH-Struktur verwendet werden.
- Setzen Sie das Feld *Format* auf MQFMT_DEAD_LETTER_HEADER, um anzugeben, dass die Daten mit einer MQDLH-Struktur beginnen.
- Legen Sie die Kontextfelder (*AccountingToken*, *ApplIdentityData* , *ApplOriginData*, *PutApplName*, *PutApplType* , *PutDate*, *PutTime*, *UserIdentifier*) fest, indem Sie eine den Umständen entsprechende Kontextoption verwenden:
  - Eine Anwendung, die eine Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht, welche mit keiner vorangegangenen Nachricht zusammenhängt, muss die Option MQPMO_DEFAULT_CONTEXT verwenden; dies bewirkt, dass der Warteschlangenmanager alle Kontextfelder im Nachrichtendeskriptor auf die zugehörigen Standardwerte setzt.
  - Eine Serveranwendung, die eine Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht, welche soeben von der Anwendung empfangen wurde, muss die Option

MQPMO_PASS_ALL_CONTEXT verwenden, damit die ursprünglichen Kontextinformationen beibehalten werden.

- Eine Serveranwendung, die eine *Antwort* auf eine Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht, welche soeben von der Anwendung empfangen wurde, muss die Option MQPMO_PASS_IDENTITY_CONTEXT verwenden; dadurch werden die Identitätsinformationen beibehalten, die Ursprungsinformationen werden jedoch auf die Angaben der Serveranwendung gesetzt.
- Ein Nachrichtenkanalagent, der eine Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht, welche er von seinem Kommunikationskanal empfangen hat, muss die Option MQPMO_SET_ALL_CONTEXT verwenden, damit die ursprünglichen Kontextinformationen beibehalten werden.

Legen Sie die Felder in der MQDLH-Struktur selbst wie folgt fest:

- Setzen Sie die Felder *CodedCharSetId*, *Encoding* und *Format* auf die Werte, die die Daten hinter der MQDLH-Struktur beschreiben. In der Regel sind dies die Werte aus dem ursprünglichen Nachrichtendeskriptor.
- Setzen Sie die Kontextfelder *PutApplType*, *PutApplName*, *PutDate*und *PutTime* auf Werte, die der Anwendung entsprechen, die die Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht. Diese Werte beziehen sich nicht auf die ursprüngliche Nachricht.
- Legen Sie in den anderen Feldern die dafür korrekten Werte fest.

Vergewissern Sie sich, dass alle Felder gültige Werte enthalten. Außerdem müssen die Zeichenfelder mit Leerzeichen bis zur definierten Länge des Felds aufgefüllt werden; die Zeichendaten dürfen nicht vorzeitig durch ein Nullzeichen beendet werden, da der Warteschlangenmanager die Null und nachfolgende Zeichen in der MQDLH-Struktur nicht in Leerzeichen konvertiert.

Nachrichten aus der Warteschlange für nicht zustellbare Nachrichten abrufen: Anwendungen, die Nachrichten aus der Warteschlange für nicht zustellbare Nachrichten abrufen, müssen überprüfen, ob die Nachrichten mit einer MQDLH-Struktur beginnen. Die Anwendung kann feststellen, ob eine MQDLH-Struktur vorhanden ist, indem sie das Feld *Format* im Nachrichtendeskriptor MQMD überprüft; enthält das Feld den Wert MQFMT_DEAD_LETTER_HEADER, beginnen die Nachrichtendaten mit einer MQDLH-Struktur. Beachten Sie außerdem, dass Nachrichten, die von Anwendungen aus der Warteschlange für nicht zustellbare Nachrichten abgerufen werden, möglicherweise abgeschnitten wurden, da sie ursprünglich für die Warteschlange zu lang waren.

Felder für MQDLH

Die MQDLH-Struktur enthält die folgenden Felder; die Felder werden in **Alphabetische Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)
Das Feld "CodedCharSetId" gibt die Zeichensatzkennung der Daten an, die durch die MQDLH-Struktur übertragen werden (für gewöhnlich handelt es sich hierbei um die Daten aus der ursprünglichen Nachricht); dieses Feld wird nicht für Zeichendaten in der MQDLH-Struktur selbst verwendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Folgende Sonderwerte sind zulässig:

MQCCSI_INHERIT
Die Zeichendaten in den Daten, die auf diese Struktur folgen, haben denselben Zeichensatz wie diese Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

Sie können MQCCSI_INHERIT nicht verwenden, wenn das Feld *PutApplType* im MQMD den Wert MQAT_BROKER enthält.

Dieser Wert wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

DestQMgrName (MQCHAR48)
Das Feld "DestQMgrName" gibt den Namen des Warteschlangenmanagers an, für den die Nachricht ursprünglich bestimmt war.

Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

DestQName (MQCHAR48)
Das Feld "DestQName" gibt den Namen der Nachrichtenwarteschlange an, für die die Nachricht ursprünglich bestimmt war.

Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

Encoding (MQLONG)
Das Feld "Encoding" gibt die numerische Codierung der Daten an, die auf die MQDLH-Struktur folgen (für gewöhnlich handelt es sich hierbei um die Daten aus der ursprünglichen Nachricht); dieses Feld wird nicht für numerische Daten in der MQDLH-Struktur selbst verwendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Felds ist `0`.

Format (MQCHAR8)
Das Feld "Format" gibt den Formatnamen der Daten an, die auf die MQDLH-Struktur folgen (für gewöhnlich handelt es sich hierbei um die Daten aus der ursprünglichen Nachricht).

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds sind mit den Regeln identisch, die für die Codierung des Felds *Format* im MQMD gelten.

Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

PutApplName (MQCHAR28)
PutApplName ist der Name der Anwendung, die die Nachricht in die Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) eingereiht hat.

Das Format des Namens hängt vom Feld *PutApplType* ab. Das Format kann je nach Release variieren. Weitere Informationen finden Sie in der Beschreibung des Felds *PutApplName* im Abschnitt „MQMD - Nachrichtendeskriptor” auf Seite 399.

Wenn der Warteschlangenmanager die Nachricht an die Warteschlange für nicht zustellbare Nachrichten umleitet, enthält das Feld *PutApplName* die ersten 28 Zeichen des Warteschlangenmanagernamens und wird gegebenenfalls mit Leerzeichen aufgefüllt.

Die Länge dieses Felds wird durch MQ_PUT_APPL_NAME_LENGTH angegeben. In der Programmiersprache C ist der Anfangswert dieses Felds eine Nullzeichenfolge. In anderen Programmiersprachen besteht dieser Anfangswert aus 28 Leerzeichen.

PutApplType (MQLONG)
PutApplType ist der Typ der Anwendung, die die Nachricht in die Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) eingereiht hat.

Dieses Feld hat die gleiche Bedeutung wie das Feld *PutApplType* im Nachrichtendeskriptor MQMD (nähere Informationen hierzu finden Sie im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399).

Wenn der Warteschlangenmanager die Nachricht an die Warteschlange für nicht zustellbare Nachrichten umleitet, enthält das Feld *PutApplType* den Wert MQAT_QMGR.

Der Anfangswert dieses Felds ist `0`.

PutDate (MQCHAR8)
PutDate ist das Datum, an dem die Nachricht in die Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) eingereiht wurde.

Das Format, das für das vom Warteschlangenmanager in diesem Feld generierte Datum verwendet wird, lautet:

- YYYYMMDD,

wobei die Zeichen Folgendes darstellen:

YYYY
Jahr (vier Ziffern)

MM
Monat des Jahres (01 bis 12)

TT
Tag des Monats (01 bis 31)

Für die *PutDate*- und *PutTime*-Felder wird in Abhängigkeit davon, dass sie Systemuhr genau auf GMT gesetzt wird, Greenwich Mean Time (GMT) verwendet.

Die Länge dieses Felds wird durch MQ_PUT_DATE_LENGTH angegeben. In der Programmiersprache C ist der Anfangswert dieses Felds eine Nullzeichenfolge. In anderen Programmiersprachen besteht dieser Anfangswert aus acht Leerzeichen.

PutTime (MQCHAR8)
PutTime ist der Zeitpunkt, zu dem die Nachricht in die Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) eingereiht wurde.

Das Format, das für die vom Warteschlangenmanager in diesem Feld generierte Uhrzeit verwendet wird, lautet:

- HHMMSSTH,

wobei die Zeichen Folgendes darstellen:

HH
Stunde (00 bis 23)

MM
Minute (00 bis 59)

SS
Sekunden (00 bis 59; siehe Anmerkung)

T
Zehntelsekunden (0 bis 9)

H
Hundertstelsekunden (0 bis 9)

Anmerkung: Wenn die Systemuhr mit einem sehr genauen Zeitstandard synchronisiert wird, wird in seltenen Fällen im Feld *PutTime* der Wert 60 oder 61 zurückgegeben. Dies geschieht, wenn Schaltsekunden in den globalen Zeitstandard aufgenommen werden.

Für die *PutDate*- und *PutTime*-Felder wird in Abhängigkeit davon, dass sie Systemuhr genau auf GMT gesetzt wird, Greenwich Mean Time (GMT) verwendet.

Die Länge dieses Felds wird durch MQ_PUT_TIME_LENGTH angegeben. In der Programmiersprache C ist der Anfangswert dieses Felds eine Nullzeichenfolge. In anderen Programmiersprachen besteht dieser Anfangswert aus acht Leerzeichen.

Ursache (MQLONG)
Das Feld "Reason" gibt an, weshalb die Nachricht nicht in die ursprüngliche Zielwarteschlange, sondern in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde.

Dieses Feld gibt die Ursache dafür an, warum die Nachricht in die Warteschlange für nicht zustellbare Nachrichten (nicht zugestellte Nachrichten) eingereiht wurde und nicht in die ursprüngliche Zielwarteschlange. Es enthält normalerweise einen der Werte des Typs MQFB_* oder MQRC_* (beispielsweise MQRC_Q_FULL). In der Beschreibung des Felds *Feedback* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399 finden Sie ausführliche Informationen zu den allgemeinen Werten des Typs MQFB_*, die ausgegeben werden können.

Liegt der Wert im Bereich MQFB_IMS_FIRST bis MQFB_IMS_LAST, kann der eigentliche IMS-Fehlercode ermittelt werden, indem MQFB_IMS_ERROR vom Wert des Felds *Reason* subtrahiert wird.

Einige Werte des Typs MQFB_* werden nur in dieses Feld geschrieben. Sie beziehen sich auf Repository-Nachrichten, Auslösenachrichten oder Übertragungswarteschlangennachrichten, die an die Warteschlange für nicht zustellbare Nachrichten weitergeleitet wurden. Diese sind:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')
Eine Anwendung, die gerade eine Auslösenachricht verarbeitet, kann die Anwendung nicht starten, welche im Feld *ApplId* der Auslösenachricht angegeben ist (siehe „MQTM - Auslösenachricht" auf Seite 589).

Unter z/OS wird die CICS-Transaktion CKTI als Beispielanwendung für die Verarbeitung von Auslösenachrichten verwendet.

MQFB_APPL_TYPE_ERROR (X'0000010B')
Eine Anwendung, die gerade eine Auslösenachricht verarbeitet, kann die Anwendung nicht starten, da das Feld *ApplType* der Auslösenachricht einen ungültigen Wert enthält (siehe „MQTM - Auslösenachricht" auf Seite 589).

Unter z/OS wird die CICS-Transaktion CKTI als Beispielanwendung für die Verarbeitung von Auslösenachrichten verwendet.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')
Die Nachricht befand sich in der Warteschlange SYSTEM.CLUSTER.TRANSMIT.QUEUE, die für eine mit der Option MQOO_BIND_ON_OPEN geöffnete Clusterwarteschlange vorgesehen war. Der ferne Clusterempfängerkanal, der für die Übertragung der Nachricht an die Zielwarteschlange verwendet werden sollte, wurde jedoch gelöscht, bevor die Nachricht gesendet werden konnte. Da MQOO_BIND_ON_OPEN angegeben wurde, kann für die Übertragung der Nachricht nur der Kanal verwendet werden, der beim Öffnen der Warteschlange ausgewählt war. Weil dieser Kanal nicht mehr verfügbar ist, wurde die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')
Die Nachricht ist keine Repository-Nachricht.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')
Die Nachricht wurde vom Exit für die automatische Kanaldefinition gestoppt.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')
Die Nachricht wurde vom Kanalnachrichtenexit gestoppt.

MQFB_TM_ERROR (X'0000010A')
Das Feld *Format* im MQMD enthält den Wert MQFMT_TRIGGER, aber die Nachricht beginnt nicht mit einer gültigen MQTM-Struktur. Beispielsweise kann die Ursache sein, dass die mnemonische Strukturkennung *StrucId* nicht gültig ist, der Wert im Feld *Version* nicht erkannt wird oder die Länge der Auslösenachricht nicht für die MQTM-Struktur ausreicht.

Unter z/OS wird die CICS-Transaktion CKTI als Beispielanwendung für die Verarbeitung von Auslösenachrichten verwendet, die diesen Rückmeldungscode generieren kann.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')
Ein Nachrichtenkanalagent hat festgestellt, dass eine Nachricht in der Übertragungswarteschlange ein falsches Format aufweist. Der Nachrichtenkanalagent reiht die Nachricht unter Verwendung dieses Rückmeldungscodes in die Warteschlange für nicht zustellbare Nachrichten ein.

Der Anfangswert dieses Felds ist MQRC_NONE.

StrucId (MQCHAR4)
Das Feld "StrucId" gibt die Struktur-ID an.

Folgende Werte sind möglich:

MQDLH_STRUC_ID
Die ID der Struktur des Headers für nicht zustellbare Nachrichten.

Für die Programmiersprache C ist auch die Konstante MQDLH_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQDLH_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Der Anfangswert dieses Felds lautet MQDLH_STRUC_ID.

Version (MQLONG)
Version ist die Strukturversionsnummer.

Folgende Werte sind möglich:

MQDLH_VERSION_1
Die Versionsnummer der Struktur des Headers für nicht zustellbare Nachrichten.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQDLH_CURRENT_VERSION
Aktuelle Version der Headerstruktur für nicht zustellbare Nachrichten

Der Anfangswert dieses Felds lautet MQDLH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQDLH

Tabelle 496. Anfangswerte der Felder in MQDLH für MQDLH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQDLH_STRUC_ID	'DLH¬'
Version	MQDLH_VERSION_1	1
Reason	MQRC_NONE	0
DestQName	--	Nullzeichenfolge oder Leerzeichen.
DestQMgrName	--	Nullzeichenfolge oder Leerzeichen.
Encoding	--	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
PutApplType	--	0
PutApplName	--	Nullzeichenfolge oder Leerzeichen.

Tabelle 496. Anfangswerte der Felder in MQDLH für MQDLH (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
PutDate	--	Nullzeichenfolge oder Leerzeichen.
PutTime	--	Nullzeichenfolge oder Leerzeichen.

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQDLH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Reason;          /* Reason message arrived on dead-letter
                                (undelivered-message) queue */
  MQCHAR48  DestQName;       /* Name of original destination queue */
  MQCHAR48  DestQMgrName;    /* Name of original destination queue
                                manager */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQDLH */
  MQLONG    CodedCharSetId;  /* Character set identifier of data that
                                follows MQDLH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQDLH */
  MQLONG    PutApplType;     /* Type of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR28  PutApplName;     /* Name of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR8   PutDate;         /* Date when message was put on dead-letter
                                (undelivered-message) queue */
  MQCHAR8   PutTime;         /* Time when message was put on the
                                dead-letter (undelivered-message)
                                queue */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQDLH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Reason         fixed bin(31), /* Reason message arrived on
                                     dead-letter (undelivered-message)
                                     queue */
  3 DestQName      char(48),      /* Name of original destination
                                     queue */
  3 DestQMgrName   char(48),      /* Name of original destination queue
                                     manager */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQDLH */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows MQDLH */
  3 Format         char(8),       /* Format name of data that follows
                                     MQDLH */
  3 PutApplType    fixed bin(31), /* Type of application that put
                                     message on dead-letter
                                     (undelivered-message) queue */
  3 PutApplName    char(28),      /* Name of application that put
                                     message on dead-letter
                                     (undelivered-message) queue */
  3 PutDate        char(8),       /* Date when message was put on
                                     dead-letter (undelivered-message)
                                     queue */
  3 PutTime        char(8);       /* Time when message was put on the
                                     dead-letter (undelivered-message)
                                     queue */
```

Deklaration in High Level Assembler

```
MQDLH                 DSECT
MQDLH_STRUCID         DS   CL4   Structure identifier
MQDLH_VERSION         DS   F     Structure version number
MQDLH_REASON          DS   F     Reason message arrived on dead-letter
*                                (undelivered-message) queue
MQDLH_DESTQNAME       DS   CL48  Name of original destination queue
MQDLH_DESTQMGRNAME    DS   CL48  Name of original destination queue
*                                manager
MQDLH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQDLH
MQDLH_CODEDCHARSETID  DS   F     Character set identifier of data that
*                                follows MQDLH
MQDLH_FORMAT          DS   CL8   Format name of data that follows MQDLH
MQDLH_PUTAPPLTYPE     DS   F     Type of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTAPPLNAME     DS   CL28  Name of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTDATE         DS   CL8   Date when message was put on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTTIME         DS   CL8   Time when message was put on the
*                                dead-letter (undelivered-message) queue
*
MQDLH_LENGTH          EQU  *-MQDLH
                      ORG  MQDLH
MQDLH_AREA            DS   CL(MQDLH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQDLH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Reason         As Long      'Reason message arrived on dead-letter'
                              '(undelivered-message) queue'
  DestQName      As String*48 'Name of original destination queue'
  DestQMgrName   As String*48 'Name of original destination queue'
                              'manager'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQDLH'
  CodedCharSetId As Long      'Character set identifier of data that'
                              'follows MQDLH'
  Format         As String*8  'Format name of data that follows MQDLH'
  PutApplType    As Long      'Type of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutApplName    As String*28 'Name of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutDate        As String*8  'Date when message was put on dead-letter'
                              '(undelivered-message) queue'
  PutTime        As String*8  'Time when message was put on the'
                              'dead-letter (undelivered-message) queue'
End Type
```

MQDMHO – Optionen zum Löschen von Nachrichtenkennungen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 497. Felder in MQDMHO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options

Überblick über MQDMHO

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Mit der **MQDMHO**-Struktur können Anwendungen Optionen festlegen, die steuern, wie Nachrichtenkennungen gelöscht werden. Die Struktur ist ein Eingabeparameter für den **MQDLTMH**-Aufruf.

Zeichensatz und Codierung: Die Daten in **MQDMHO** müssen im Zeichensatz und in der Codierung der Anwendung (**MQENC_NATIVE**) enthalten sein.

Felder für MQDMHO

Die MQDMHO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)

Folgende Werte sind möglich:

MQDMHO_NONE

Keine Optionen angegeben.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQDMHO_NONE**.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQDMHO_STRUC_ID

ID der Struktur zum Löschen von Optionen für Nachrichtenkennungen.

Für die Programmiersprache C wird außerdem die Konstante **MQDMHO_STRUC_ID_ARRAY** definiert; diese hat denselben Wert wie **MQDMHO_STRUC_ID**, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQDMHO_STRUC_ID**.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQDMHO_VERSION_1
Version-1 der Optionsstruktur für Nachrichtenkennungen.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQDMHO_CURRENT_VERSION
Aktuelle Version der Optionsstruktur zum Löschen von Nachrichtenkennungen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQDMHO_VERSION_1**.

Anfangswerte und Sprachendeklarationen für MQDMHO

Tabelle 498. Anfangswerte der Felder in MQDMHO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQDMHO_STRUC_ID	'DMHO'
Version	MQDMHO_VERSION_1	1
Options	MQDMHO_NONE	0

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQDMHO_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQDLTMH */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQDMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQDLTMH */
```

Deklaration in High Level Assembler

```
MQDMHO                DSECT
MQDMHO_STRUCID        DS   CL4   Structure identifier
MQDMHO_VERSION        DS   F     Structure version number
MQDMHO_OPTIONS        DS   F     Options that control the action of
*                                MQDLTMH
MQDMHO_LENGTH         EQU  *-MQDMHO
MQDMHO_AREA           DS   CL(MQDMHO_LENGTH)
```

MQDMPO – Optionen für das Löschen von Nachrichteneigenschaften

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. MQDMPO-Struktur – Optionen für Nachrichteneigenschaften löschen

Tabelle 499. Felder in MQDMPO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQDMPO	Options

Überblick über MQDMPO

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Mit der MQDMPO-Struktur können Anwendungen Optionen festlegen, die steuern, wie Nachrichteneigenschaften gelöscht werden. Bei der Struktur handelt es sich um einen Eingabeparameter im MQDLTMP-Aufruf.

Zeichensatz und Codierung: Die Daten in MQDMPO müssen im Zeichensatz und in der Codierung der Anwendung (MQENC_NATIVE) enthalten sein.

Felder für MQDMPO

Optionsstruktur zum Löschen von Nachrichteneigenschaften - Felder

Die MQDMPO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Feld Options

Positionsoptionen: Die folgenden Optionen beziehen sich auf die relative Position der Eigenschaft verglichen mit dem Eigenschaftscursor.

MQDMPO_DEL_FIRST
Löscht die erste mit dem angegebenen Namen übereinstimmende Eigenschaft.

MQDMPO_DEL_PROP_UNDER_CURSOR
Löscht die Eigenschaft, auf die der Eigenschaftscursor verweist, d. h. die Eigenschaft, die zuletzt über die Option MQIMPO_INQ_FIRST oder MQIMPO_INQ_NEXT abgefragt wurde.

Der Eigenschaftscursor wird zurückgesetzt, wenn die Nachrichtenkennung wiederverwendet wird. Er wird ebenfalls zurückgesetzt, wenn die Nachrichtenkennung im Feld *MsgHandle* der MQGMO-Struktur bei einem MQGET-Aufruf oder einer MQPMO-Struktur eines MQPUT-Aufrufs angegeben wird.

Wird diese Option verwendet, wenn der Eigenschaftscursor noch nicht eingerichtet ist, schlägt der Aufruf mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl. Wurde die Eigenschaft, auf die der Eigenschaftscursor verweist, bereits gelöscht, schlägt der Aufruf ebenfalls mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl.

Ist keine dieser Optionen erforderlich, kann die folgende Option verwendet werden:

MQDMPO_NONE
Keine Optionen angegeben.

Dieses Feld ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQDMPO_DEL_FIRST.

StrucId (MQCHAR4)
Optionsstruktur zum Löschen von Nachrichteneigenschaften- Feld StrucId

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQDMPO_STRUC_ID
ID der Struktur von Optionen zum Löschen von Nachrichteneigenschaften.

Für die Programmiersprache C wird auch die Konstante MQDMPO_STRUC_ID_ARRAY definiert; diese hat denselben Wert wie MQDMPO_STRUC_ID, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQDMPO_STRUC_ID.

Version (MQLONG)
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Feld Version

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQDMPO_VERSION_1
Versionsnummer der Struktur von Optionen zum Löschen von Nachrichteneigenschaften.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQDMPO_CURRENT_VERSION
Aktuelle Version der Optionsstruktur zum Löschen von Nachrichteneigenschaften.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQDMPO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQDMPO

Optionsstruktur zum Löschen von Nachrichteneigenschaften - Anfangswerte

Tabelle 500. Anfangswerte der Felder in MQDPMO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQDMPO_STRUC_ID	'DMPO'
Version	MQDMPO_VERSION_1	1
Options	Optionen zur Steuerung der Aktion von MQDLTMP	MQDMPO_NONE

Tabelle 500. Anfangswerte der Felder in MQDPMO (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQDMPO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

Deklaration in Programmiersprache C
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Deklaration für Programmiersprache C

```
typedef struct tagMQDMPO MQDMPO;
struct tagMQDMPO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQDLTMP */
};
```

COBOL-Delaration
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Deklaration für Programmiersprache COBOL

```
**   MQDMPO structure
  10 MQDMPO.
**    Structure identifier
   15 MQDMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQDMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQDLTMP
   15 MQDMPO-OPTIONS               PIC S9(9) BINARY.
```

Deklaration in PL/I
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Deklaration für Programmiersprache PL/I

```
Dcl
 1 MQDPMO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQDLTMP */
```

Deklaration in High Level Assembler
Optionsstruktur zum Löschen von Nachrichteneigenschaften - Deklaration für Assemblersprache

```
MQDMPO                   DSECT
MQDMPO_STRUCID           DS   CL4  Structure identifier
MQDMPO_VERSION           DS   F    Structure version number
MQDMPO_OPTIONS           DS   F    Options that control the
*                                  action of MQDLTMP
MQDMPO_LENGTH            EQU  *-MQDMPO
MQDMPO_AREA              DS   CL(MQDMPO_LENGTH)
```

MQEPH - Eingebetteter PCF-Header

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 501. Felder in MQEPH		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
StrucLength	Länge der MQEPH-Struktur plus MQCFH-Struktur und Parameterstrukturen, die darauf folgen	StrucLength
Encoding	Numerische Codierung der Daten, die auf die letzte PCF-Parameterstruktur folgen	Encoding
CodedCharSetId	Zeichensatzkennung der Daten, die auf die letzte PCF-Parameterstruktur folgen	CodedCharSetId
Format	Formatname der Daten, die auf die letzte PCF-Parameterstruktur folgen	Format
Flags	Markierungen	Flags
PCFHeader	PCF-Header (Programmable Command Format)	PCFHeader

Übersicht über MQEPH

Verfügbarkeit: Alle WebSphere MQ-Plattformen.

Zweck: Die MQEPH-Struktur beschreibt die zusätzlichen Daten, die in einer PCF-Nachricht vorhanden sind. Das Feld *PCFHeader* definiert die PCF-Parameter, die auf diese Struktur folgen. Dadurch können Sie hinter den PCF-Nachrichtendaten weitere Header einfügen.

Formatname: MQFMT_EMBEDDED_PCF

Zeichensatz und Codierung: Die Daten in MQEPH müssen den Zeichensatz haben, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und die Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE festgelegt wird.

Legen Sie an folgenden Stellen den Zeichensatz und die Codierung der MQEPH-Struktur in den Feldern *CodedCharSetId* und *Encoding* fest:

- In MQMD (wenn sich die MQEPH-Struktur am Anfang der Nachrichtendaten befindet) oder
- In der Headerstruktur, die der MQEPH-Struktur vorausgeht (alle anderen Fälle).

Verwendung: MQEPH-Strukturen können nicht für das Senden von Befehlen an den Befehlsserver oder an einen anderen Server gesendet werden, der das Programmable Command Format für Warteschlangenmanager akzeptiert.

Analog hierzu werden vom Befehlsserver oder einem anderen Server, der das Programmable Command Format für Warteschlangenmanager akzeptiert, keine Antworten oder Ereignisse generiert, die MQEPH-Strukturen enthalten.

Felder für MQEPH

Die MQEPH-Struktur enthält die folgenden Felder, die in **alphabetischer Reihenfolge** beschrieben werden:

CodedCharSetId (MQLONG)

Dieses Feld enthält die Zeichensatzkennung der Daten, die auf die MQEPH-Struktur und die zugehörigen PCF-Parameter folgen; es wird nicht für die Zeichendaten in der MQEPH-Struktur selbst verwendet.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

Encoding (MQLONG)

Dieses Feld enthält die numerische Codierung der Daten, die auf die MQEPH-Struktur und die zugehörigen PCF-Parameter folgen; es wird nicht für die Zeichendaten in der MQEPH-Struktur selbst verwendet.

Der Anfangswert dieses Feldes ist 0.

Flags (MQLONG)

Die folgenden Werte sind verfügbar:

MQEPH_NONE
Es wurden keine Flags angegeben. MQEPH_NONE wird zur Unterstützung der Programmdokumentation definiert. Diese Konstante ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

MQEPH_CCSID_EMBEDDED
Der Zeichensatz der Parameter, die Zeichendaten enthalten, wird jeweils im Feld "CodedCharSetId" in der jeweiligen Struktur angegeben. Der Zeichensatz der Felder "StrucId" und "Format" ist im Feld "CodedCharSetId" in der Headerstruktur angegeben, die der MQEPH-Struktur vorangeht, oder im Feld "CodedCharSetId" im MQMD, wenn sich die MQEPH-Struktur am Anfang der Nachricht befindet.

Der Anfangswert dieses Felds lautet MQEPH_NONE.

Format (MQCHAR8)

Dieses Feld gibt den Formatnamen der Daten an, die auf die MQEPH-Struktur und die zugehörigen PCF-Parameter folgen.

Der Anfangswert dieses Felds ist MQFMT_NONE.

PCFHeader (MQCFH)

Dies ist der PCF-Header (Programmable Command Format), der die PCF-Parameter definiert, die der MQEPH-Struktur folgen. So können Sie den PCF-Nachrichtendaten mit anderen Headern folgen.

Der PCF-Header ist anfänglich mit den folgenden Werten definiert:

Tabelle 502. Anfangswerte der Felder in MQCFH

Name des Felds	Name der Konstante	Wert der Konstanten
Type	MQCFT_NONE	0
StrucLength	MQCFH_STRUC_LENGH	36
Version	MQCFH_VERSION_3	3
StrucLength	--	0
Command	MQCMD_NONE	0
MsgSeqNumber	--	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	--	0

Die Anwendung muss den Wert MQCFT_NONE im Feld Type in einen gültigen Strukturtyp ändern, der sich für den Zweck eignet, zu dem der eingebettete PCF-Header genutzt wird.

StrucId (MQCHAR4)

Folgende Werte sind möglich:

MQEPH_STRUC_ID
Die ID für die Struktur des Verteilungsheaders.

Für die Programmiersprache C ist auch die Konstante MQEPH_STRUC_ID_ARRAY definiert, die denselben Wert wie die Konstante MQDH_STRUC_ID hat. Allerdings handelt es sich dabei nicht um eine Zeichenfolge, sondern um eine Feldgruppe von Zeichen.

Der Anfangswert dieses Felds lautet MQEPH_STRUC_ID.

StrucLength (MQLONG)

Dies ist das Datenvolumen, das der nächsten Headerstruktur vorangeht. Die Angabe umfasst Folgendes:

- Die Länge des MQEPH-Headers
- Die Länge aller PCF-Parameter, die auf den Header folgen
- Aufgefüllte Leerzeichen hinter diesen Parametern

StrucLength muss ein Vielfaches von 4 sein.

Der Strukturbereich mit fester Länge wird durch MQEPH_STRUC_LENGTH_FIXED definiert.

Der Anfangswert dieses Felds ist 68.

Version (MQLONG)

Folgende Werte sind möglich:

MQEPH_VERSION_1
Versionsnummer für die integrierte PCF-Headerstruktur.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCFH_VERSION_3
Aktuelle Version der eingebetteten PCF-Headerstruktur.

Der Anfangswert dieses Felds lautet MQEPH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQEPH

Tabelle 503. Anfangswerte der Felder in MQEPH für MQEPH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQEPH_STRUC_ID	'EPH¬'
Version	MQEPH_VERSION_1	1
StrucLength	MQEPH_STRUC_LENGTH_FIXED	68
Encoding	--	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQEPH_NONE	0
PCFHeader	Namen und Werte gemäß der Definition in Tabelle 502 auf Seite 344	0

Tabelle 503. Anfangswerte der Felder in MQEPH für MQEPH (Forts.)		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQEPH_DEFAULT enthält die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Total length of MQEPH including the MQCFH
                                and parameter structures that follow it */
  MQLONG   Encoding;         /* Numeric encoding of data that follows last
                                PCF parameter structure */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows last PCF parameter structure */
  MQCHAR8  Format;           /* Format name of data that follows last PCF
                                parameter structure */
  MQLONG   Flags;            /* Flags */
  MQCFH    PCFHeader;        /* Programmable command format header */
 };
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQEPH based,
  3 StrucId         char(4),       /* Structure identifier */
  3 Version         fixed bin(31), /* Structure version number */
  3 StrucLength     fixed bin(31), /* Total Length of MQEPH including the
                                      MQCFH and parameter structures that
                                      follow it
  3 Encoding        fixed bin(31), /* Numeric encoding of data that follows
                                      last PCF parameter structure
  3 CodedCharSetId  fixed bin(31), /* Character set identifier of data that
                                      follows last PCF parameter structure
  3 Format          char(8),       /* Format name of data that follows last
                                      PCF parameter structure */
  3 Flags           fixed bin(31), /* Flags */
  3 PCFHeader,                     /* Programmable command format header
   5 Type           fixed bin(31), /* Structure type */
   5 StrucLength    fixed bin(31), /* Structure length */
   5 Version        fixed bin(31), /* Structure version number */
   5 Command        fixed bin(31), /* Command identifier */
   5 MsgseqNumber   fixed bin(31), /* Message sequence number */
   5 Control        fixed bin(31), /* Control options */
   5 CompCode       fixed bin(31), /* Completion code */
   5 Reason         fixed bin(31), /* Reason code qualifying completion code */
   5 ParameterCount fixed bin(31); /* Count of parameter structures */
```

Deklaration in High Level Assembler

```
MQEPH                           DSECT
MQEPH_STRUCID                   DS   CL4   Structure identifier
MQEPH_VERSION                   DS   F     Structure version number
MQEPH_STRUCLENGTH               DS   F     Total length of MQEPH including the
*                                          MQCFH and parameter structures that
                                           follow it
MQEPH_ENCODING                  DS   F     Numeric encoding of data that follows
*                                          last PCF parameter structure
MQEPH_CODEDCHARSETID            DS   F     Character set identifier of data that
*                                          follows last PCF parameter structure
MQEPH_FORMAT                    DS   CL8   Format name of data that follows last
*                                          PCF parameter structure
MQEPH_FLAGS                     DS   F     Flags
MQEPH_PCFHEADER                 DS   0F    Force fullword alignment
MQEPH_PCFHEADER_TYPE            DS   F     Structure type
MQEPH_PCFHEADER_STRUCLENGTH     DS   F     Structure length
MQEPH_PCFHEADER_VERSION         DS   F     Structure version number
MQEPH_PCFHEADER_COMMAND         DS   F     Command identifier
MQEPH_PCFHEADER_MSGSEQNUMBER    DS   F     Structure length
MQEPH_PCFHEADER_CONTROL         DS   F     Control options
MQEPH_PCFHEADER_COMPCODE        DS   F     Completion code
MQEPH_PCFHEADER_REASON          DS   F     Reason code qualifying completion code
MQEPH_PCFHEADER_PARAMETER COUNT DS   F     Count of parameter structures
MQEPH_PCFHEADER_LENGTH          EQU  *-MQEPH_PCFHEADER
                                ORG  MQEPH_PCFHEADER
MQEPH_PCFHEADER_AREA            DS   CL(MQEPH_PCFHEADER_LENGTH)
*
MQEPH_LENGTH                    EQU  *-MQEPH
                                ORG  MQEPH
MQEPH_AREA                      DS   CL(MQEPH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQEPH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
  StrucLength     As Long     'Total length of MQEPH structure including the MQCFH'
                              'and parameter structures that follow it'
  Encoding        As Long     'Numeric encoding of data that follows last'
```

```
                              'PCF parameter structure'
  CodedCharSetId  As Long     'Character set identifier of data that'
                              'follows last PCF parameter structure'
  Format          As String*8 'Format name of data that follows last PCF'
                              'parameter structure'
  Flags           As Long     'Flags'
  PCFHeader       As MQCFH    'Programmable command format header'
 End Type

Global MQEPH_DEFAULT As MQEPH
```

MQGMO – Nachrichtenabrufoptionen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 504. Felder in MQGMO		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQGET	MQGMO - Options-Feld
WaitInterval	Warteintervall	WaitInterval
Signal1	Signal	Signal1
Signal2	Signal-ID	Signal2
ResolvedQName	Aufgelöster Name der Zielwarteschlange	ResolvedQName
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.		
MatchOptions	Optionen zur Steuerung der für MQGET verwendeten Auswahlbedingungen	MatchOptions
GroupStatus	Flag, das angibt, ob sich die abgerufene Nachricht in einer Gruppe befindet	GroupStatus
SegmentStatus	Flag, das angibt, ob die abgerufene Nachricht ein Segment einer logischen Nachricht ist	SegmentStatus
Segmentation	Flag, das angibt, ob für die abgerufene Nachricht eine weitere Segmentierung zulässig ist	Segmentation
Reserved1	Reserved	Reserved1
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQGMO_VERSION_3 ist.		
MsgToken	Nachrichtentoken	MsgToken
ReturnedLength	Länge der zurückgegebenen Nachrichtendaten (Bytes)	ReturnedLength
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQGMO_VERSION_4 ist.		
Reserved2	Reserved	Reserved2
MsgHandle	Die Kennung für eine Nachricht, die mit den Eigenschaften der Nachricht belegt wird, die aus der Warteschlange abgerufen wird.	MsgHandle

Übersicht über MQGMO

Verfügbarkeit: Alle WebSphere MQ-Plattformen.

Zweck: Mithilfe der MQGMO-Struktur kann die Anwendung steuern, wie Nachrichten aus Warteschlangen entfernt werden. Die Struktur ist ein Ein-/Ausgabeparameter für den MQGET-Aufruf.

Version: Die aktuelle Version von MQGMO ist MQGMO_VERSION_4. Bestimmte Felder sind nur in bestimmten Versionen von MQGMO verfügbar. Wenn Sie Anwendungen zwischen mehreren Umgebungen portieren müssen, müssen Sie sicherstellen, dass in allen Umgebungen die gleiche MQGMO-Version verwendet wird. Informationen dazu, welche Felder nur in bestimmten Versionen der Struktur vorliegen, finden Sie im Abschnitt „MQGMO – Nachrichtenabrufoptionen" auf Seite 348 und in den Feldbeschreibungen.

Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützten Programmiersprachen bereitgestellt werden, enthalten die aktuellste Version von MQGMO, die von der Umgebung unterstützt wird, wobei der Anfangswert des Feldes *Version* auf MQGMO_VERSION_1 gesetzt ist. Um Felder zu verwenden, die in der Struktur der Version 1 nicht verfügbar sind, setzen Sie das Feld *Version* auf die Versionsnummer der erforderlichen Version.

Zeichensatz und Codierung: Die Daten in MQGMO müssen den Zeichensatz haben, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und die Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE festgelegt wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQGMO

Die MQGMO-Struktur enthält die folgenden Felder, die in **alphabetischer Reihenfolge** beschrieben werden:

GroupStatus (MQCHAR)
Dieses Flag gibt an, ob die abgerufene Nachricht in einer Gruppe enthalten ist.

Es entspricht einem der folgenden Werte:

MQGS_NOT_IN_GROUP
Nachricht befindet sich nicht in einer Gruppe.

MQGS_MSG_IN_GROUP
Nachricht befindet sich in einer Gruppe, ist jedoch nicht die letzte in der Gruppe.

MQGS_LAST_MSG_IN_GROUP
Die Nachricht ist die letzte in der Gruppe.

Dieser Wert wird auch zurückgegeben, wenn die Gruppe lediglich aus einer Nachricht besteht.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Felds lautet MQGS_NOT_IN_GROUP. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.

MatchOptions (MQLONG)

Mit diesen Optionen kann die Anwendung auswählen, welche Felder im Parameter *MsgDesc* verwendet werden sollen, um die vom MQGET-Aufruf zurückgegebene Nachricht auszuwählen. Die Anwendung legt in diesem Feld die erforderlichen Optionen fest und setzt anschließend die entsprechenden Felder im Parameter *MsgDesc* auf die Werte, die für diese Felder erforderlich sind. Nur Nachrichten, die diese Werte im MQMD für diese Nachricht aufweisen, können mit dem Parameter *MsgDesc* im MQGET-Aufruf abgerufen werden. Felder, für die die entsprechende Abgleichoption *nicht* angegeben ist, werden bei der Auswahl der zurückzugebenden Nachricht ignoriert. Wenn Sie im MQGET-Aufruf keine Auswahlkriterien angeben (d. h., *jede* Nachricht ist zulässig), setzen Sie *MatchOptions* auf MQMO_NONE.

- Unter z/OS sind die verwendbaren Auswahlkriterien möglicherweise durch den Indextyp beschränkt, der für die Warteschlange genutzt wird. Weitere Details finden Sie im Warteschlangenattribut *IndexType* .

Wenn Sie MQGMO_LOGICAL_ORDER angeben, kommen nur bestimmte Nachrichten für die Rückgabe durch den nächsten MQGET-Aufruf infrage:

- Wenn keine aktuelle Gruppe oder logische Nachricht vorhanden ist, können nur Nachrichten mit *MsgSeqNumber* gleich 1 und *Offset* gleich 0 zurückgegeben werden. In dieser Situation können

Sie eine oder mehrere der folgenden Abgleichoptionen für die Auswahl der zurückzugebenden infrage kommenden Nachrichten verwenden:

  - MQMO_MATCH_MSG_ID
  - MQMO_MATCH_CORREL_ID
  - MQMO_MATCH_GROUP_ID

- Ist eine aktuelle Gruppe oder logische Nachricht *vorhanden*, kommt nur die nächste Nachricht in der Gruppe oder das nächste Segment in der logischen Nachricht für die Rückgabe infrage. Dies kann nicht durch die Angabe von Optionen des Typs MQMO_* geändert werden.

In beiden oben genannten Fällen können Sie Abgleichoptionen angeben, die nicht anwendbar sind. Der Wert des relevanten Felds im Parameter *MsgDesc* muss jedoch mit dem Wert des entsprechenden Felds in der zurückzugebenden Nachricht übereinstimmen. Der Aufruf schlägt mit dem Ursachencode MQRC_MATCH_OPTIONS_ERROR fehl, wenn diese Bedingung nicht erfüllt ist.

MatchOptions wird ignoriert, wenn Sie entweder MQGMO_MSG_UNDER_CURSOR oder MQGMO_BROWSE_MSG_UNDER_CURSOR angeben.

Der Abruf von Nachrichten auf Grundlage der Nachrichteneigenschaft erfolgt nicht über Abgleichoptionen; der Abschnitt „SelectionString (MQCHARV)" auf Seite 472 enthält weitere Informationen hierzu.

Sie können eine oder mehrere der folgenden Abgleichoptionen angeben:

MQMO_MATCH_MSG_ID
Die abzurufende Nachricht muss eine Nachrichten-ID aufweisen, die dem Wert des Feldes *MsgId* im Parameter *MsgDesc* des MQGET-Aufrufs entspricht. Dieser Abgleich erfolgt zusätzlich zu eventuell anderen anwendbaren Abgleichen (beispielsweise der Korrelations-ID).

Wenn Sie diese Option nicht angeben, wird das Feld *MsgId* im Parameter *MsgDesc* ignoriert und alle Nachrichten-IDs stimmen überein.

Anmerkung: Die Nachrichten-ID MQMI_NONE ist ein Sonderwert, bei dem *jede* Nachrichten-ID im MQMD für die Nachricht als Übereinstimmung gilt. Daher führt die Angabe von MQMO_MATCH_MSG_ID mit MQMI_NONE zu dem Ergebnis, das auch erzielt wird, wenn Sie MQMO_MATCH_MSG_ID *nicht* angeben.

MQMO_MATCH_CORREL_ID
Die abzurufende Nachricht muss eine Korrelations-ID haben, die dem Wert des Felds *CorrelId* im Parameter *MsgDesc* des MQGET-Aufrufs entspricht. Dieser Abgleich erfolgt zusätzlich zu eventuell anderen anwendbaren Abgleichen (beispielsweise der Nachrichten-ID).

Wenn Sie diese Option nicht angeben, wird das Feld *CorrelId* im Parameter *MsgDesc* ignoriert und alle Korrelations-IDs stimmen überein.

Anmerkung: Die Korrelations-ID MQCI_NONE ist ein Sonderwert, bei dem *jede* Korrelations-ID im MQMD für die Nachricht als Übereinstimmung gilt. Daher führt die Angabe von MQMO_MATCH_CORREL_ID mit MQCI_NONE zu dem Ergebnis, das auch erzielt wird, wenn Sie MQMO_MATCH_CORREL_ID *nicht* angeben.

MQMO_MATCH_GROUP_ID
Die abzurufende Nachricht muss eine Gruppen-ID haben, die dem Wert des Felds *GroupId* im Parameter *MsgDesc* des MQGET-Aufrufs entspricht. Dieser Abgleich erfolgt zusätzlich zu eventuell anderen anwendbaren Abgleichen (beispielsweise der Korrelations-ID).

Wenn Sie diese Option nicht angeben, wird das Feld *GroupId* im Parameter *MsgDesc* ignoriert und alle Gruppen-IDs stimmen überein.

Anmerkung: Die Gruppen-ID MQGI_NONE ist ein Sonderwert, bei dem *jede* Gruppen-ID im MQMD für die Nachricht als Übereinstimmung gilt. Daher führt die Angabe von MQMO_MATCH_GROUP_ID mit MQGI_NONE zu dem Ergebnis, das auch erzielt wird, wenn Sie MQMO_MATCH_GROUP_ID *nicht* angeben.

MQMO_MATCH_MSG_SEQ_NUMBER
Die abzurufende Nachricht muss eine Nachrichtenfolgenummer aufweisen, die dem Wert des Felds *MsgSeqNumber* im Parameter *MsgDesc* des MQGET-Aufrufs entspricht. Dieser Abgleich erfolgt zusätzlich zu eventuell anderen anwendbaren Abgleichen (beispielsweise der Gruppen-ID).

Wenn Sie diese Option nicht angeben, wird das Feld *MsgSeqNumber* im Parameter *MsgDesc* ignoriert und alle Nachrichtenfolgenummern stimmen überein.

MQMO_MATCH_OFFSET
Die abzurufende Nachricht muss eine relative Position aufweisen, die dem Wert des Feldes *Offset* im Parameter *MsgDesc* des MQGET-Aufrufs entspricht. Dieser Abgleich erfolgt zusätzlich zu eventuell anderen anwendbaren Abgleichen (beispielsweise der Nachrichtenfolgenummer).

Wenn Sie diese Option nicht angeben, wird das Feld *Offset* im Parameter *MsgDesc* ignoriert und alle Offsets stimmen überein.

- Diese Option wird unter z/OS nicht unterstützt.

MQMO_MATCH_MSG_TOKEN
Die abzurufende Nachricht muss ein Nachrichtentoken aufweisen, das dem Wert des Feldes *MsgToken* in der MQGMO-Struktur entspricht, die im MQGET-Aufruf angegeben ist.

Sie können diese Option für alle lokalen Warteschlangen angeben. Wenn Sie es für eine Warteschlange angeben, deren *IndexType* MQIT_MSG_TOKEN (eine WLM-verwaltete Warteschlange) ist, können Sie mit MQMO_MATCH_MSG_TOKEN keine anderen Abgleichoptionen angeben.

Sie können MQMO_MATCH_MSG_TOKEN nicht in Verbindung mit MQGMO_WAIT oder MQGMO_SET_SIGNAL angeben. Wenn die Anwendung warten möchte, bis eine Nachricht in einer Warteschlange eintrifft, die den Wert MQIT_MSG_TOKEN im Feld *IndexType* aufweist, geben Sie MQMO_NONE an.

Wenn Sie diese Option übergehen, wird das Feld *MsgToken* in der MQGMO-Struktur ignoriert. In diesem Fall gilt jedes Nachrichtentoken als Übereinstimmung.

Werden keine der oben beschriebenen Optionen angegeben, können Sie die folgende Option verwenden:

MQMO_NONE
Bei der Auswahl der zurückzugebenden Nachrichten findet kein Abgleich statt; alle Nachrichten in der Warteschlange kommen für den Abruf infrage (dies unterliegt jedoch der Steuerung durch die Optionen MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE und MQGMO_COMPLETE_MSG).

MQMO_NONE dient zur Unterstützung der Programmdokumentation. Diese Option ist nicht für die Verwendung mit anderen Optionen des Typs MQMO_* vorgesehen. Da sie jedoch den Wert null hat, kann eine derartige Verwendung nicht erkannt werden.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds lautet MQMO_MATCH_MSG_ID mit MQMO_MATCH_CORREL_ID. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.

Anmerkung: Der Anfangswert des Felds *MatchOptions* ist für die Kompatibilität mit früheren MQSeries -Warteschlangenmanagern definiert. Beim Lesen einer Reihe von Nachrichten aus einer Warteschlange ohne Verwendung von Auswahlkriterien erfordert dieser Anfangswert jedoch, dass die Anwendung die Felder *MsgId* und *CorrelId* vor jedem MQGET-Aufruf auf MQMI_NONE und MQCI_NONE zurücksetzt. Vermeiden Sie es, *MsgId* und *CorrelId* zurückzusetzen, indem Sie *Version* auf MQGMO_VERSION_2und *MatchOptions* auf MQMO_NONE setzen.

MsgHandle (MQHMSG)

Wenn die Option MQGMO_PROPERTIES_AS_Q_DEF angegeben wird und das Warteschlangenattribut PropertyControl nicht auf MQPROP_FORCE_MQRFH2 gesetzt ist, ist dies die Kennung für eine Nachricht, die mit den Eigenschaften der aus der Warteschlange abgerufenen Nachricht gefüllt wird. Die Ken-

nung wird durch einen MQCRTMH-Aufruf erzeugt. Alle der Kennung bereits zugeordneten Eigenschaften werden vor dem Abrufen einer Nachricht gelöscht.

Auch der folgende Wert kann angegeben werden:
MQHM_NONE
  Keine Nachrichtenkennung angegeben

Für den MQGET-Aufruf ist kein Nachrichtendeskriptor erforderlich, wenn eine gültige Nachrichtenkennung für die Übernahme der Nachrichteneigenschaften angegeben und in der Ausgabe verwendet wird. In diesem Fall wird der der Nachrichtenkennung zugewiesene Nachrichtendeskriptor für Eingabefelder verwendet.

Wenn für den MQGET-Aufruf ein Nachrichtendeskriptor angegeben wird, hat der immer Vorrang vor dem der Nachrichtenkennung zugewiesenen Nachrichtendeskriptor.

Wenn MQGMO_PROPERTIES_FORCE_MQRFH2 angegeben ist oder wenn eine MQGMO_PROPERTIES_AS_Q_DEF angegeben ist und das Warteschlangenattribut `PropertyControl` den Wert MQPROP_FORCE_MQRFH2 hat, schlägt der Aufruf mit dem Ursachencode MQRC_MD_ERROR fehl, wenn kein Nachrichtendeskriptorparameter angegeben ist.

Bei der Rückgabe eines MQGET-Aufrufs werden die dieser Nachrichtenkennung zugewiesenen Eigenschaften und der Nachrichtendeskriptor aktualisiert, um den Status der abgerufenen Nachricht wiederzugeben (sowie den Nachrichtendeskriptor, wenn im MQGET-Aufruf angegeben). Die Eigenschaften der Nachricht können danach mit dem MQINQMP-Aufruf abgefragt werden.

Außer gegebenenfalls bei Erweiterungen des Nachrichtendeskriptors ist eine Eigenschaft, die mit dem Aufruf MQINQMP abgefragt werden kann, nicht in den Nachrichtendaten enthalten. Wenn eine Nachricht in der Warteschlange Eigenschaften in den Nachrichtendaten enthält, werden diese vor der Rückgabe der Daten an die Anwendung aus den Nachrichtendaten gelöscht.

Wird keine Nachrichtenkennung bereitgestellt oder eine kleinere Version als MQGMO_VERSION_4 verwendet, müssen Sie im MQGET-Aufruf einen gültigen Nachrichtendeskriptor angeben. Alle vorhandenen Nachrichteneigenschaften (außer denen im Nachrichtendeskriptor) werden abhängig vom Wert der Eigenschaftsoptionen in der Struktur MQGMO und dem Warteschlangenattribut `PropertyControl` zurückgemeldet.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQHM_NONE. Dieses Feld wird ignoriert, wenn `Version` kleiner ist als MQGMO_VERSION_4.

MsgToken (MQBYTE16)
Feld "MsgToken" - MQGMO-Struktur. Dieses Feld wird vom Warteschlangenmanager für die eindeutige Identifizierung einer Nachricht verwendet.

Hierbei handelt es sich um eine Bytefolge, die vom Warteschlangenmanager für die eindeutige Identifizierung einer Nachricht in einer Warteschlange generiert wird. Das Nachrichtentoken wird generiert, wenn die Nachricht das erste Mal in den Warteschlangenmanager gestellt wird. Sofern der Warteschlangenmanager nicht erneut gestartet wird, bleibt es an die Nachricht gebunden, bis diese permanent aus dem Warteschlangenmanager entfernt wird.

Sobald die Nachricht aus der Warteschlange entfernt wird, verliert der Wert von *MsgToken*, über den die Instanz der Nachricht identifiziert wurde, seine Gültigkeit und wird niemals wiederverwendet. Wenn der Warteschlangenmanager erneut gestartet wird, kann es vorkommen, dass das im Feld *MsgToken* angegebene Nachrichtentoken, mit dem eine Nachricht in der Warteschlange identifiziert wurde, nach dem Neustart nicht mehr gültig ist. Das im Feld *MsgToken* angegebene Nachrichtentoken wird jedoch niemals für die Identifizierung einer anderen Nachrichteninstanz wiederverwendet. Das Nachrichtentoken im Feld *MsgToken* wird vom Warteschlangenmanager generiert und ist für keine externe Anwendung sichtbar.

Wird eine Nachricht bei einer MQGMO-Struktur der Version 3 oder höher aufgrund eines MQGET-Aufrufs zurückgegeben, wird vom Warteschlangenmanager in der MQGMO-Struktur der Wert im Feld *MsgToken* zurückgegeben, über den die Nachricht in der Warteschlange identifiziert wird. Es gibt jedoch eine Ausnahme: Wird die Nachricht außerhalb eines Synchronisationspunkts aus der Warteschlange entfernt, gibt der Warteschlangenmanager möglicherweise kein Nachrichtentoken im Feld *MsgToken* zurück, da

die Identifizierung der zurückgegebenen Nachricht in einem nachfolgenden MQGET-Aufruf nicht sinnvoll wäre. Anwendungen sollten das Feld *MsgToken* nur für den Verweis auf die Nachricht in nachfolgenden MQGET-Aufrufen verwenden.

Wenn im Feld *MsgToken* ein Nachrichtentoken bereitgestellt wurde und im Feld *MatchOption* die Abgleichoption MQMO_MATCH_MSG_TOKEN ohne Festlegung von MQGMO_MSG_UNDER_CURSOR oder MQGMO_BROWSE_MSG_UNDER_CURSOR angegeben ist, kann nur die Nachricht zurückgegeben werden, die durch den betreffenden *MsgToken*-Wert identifiziert wird. Die Option ist unabhängig vom INDXTYPE-Wert bei allen lokalen Warteschlangen gültig; unter z/OS darf die Option INDXTYPE(MSGTOKEN) nur für Workload Manager-Warteschlangen verwendet werden.

Alle anderen unter *MatchOptions* angegebenen Abgleichoptionen werden überprüft. Falls keine Übereinstimmung gefunden wird, wird MQRC_NO_MSG_AVAILABLE zurückgegeben. Wird MQGMO_BROWSE_NEXT mit MQMO_MATCH_MSG_TOKEN codiert, wird die durch *MsgToken* identifizierte Nachricht nur zurückgegeben, wenn sie hinter dem Anzeigecursor für die aufrufende Kennung liegt.

Ist MQGMO_MSG_UNDER_CURSOR oder MQGMO_BROWSE_MSG_UNDER_CURSOR angegeben, wird MQMO_MATCH_MSG_TOKEN ignoriert.

MQMO_MATCH_MSG_TOKEN ist bei den folgenden Nachrichtenabrufoptionen nicht gültig:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

Bei einem MQGET-Aufruf mit der Angabe MQMO_MATCH_MSG_TOKEN muss dem Aufruf eine MQGMO-Struktur der Version 3 oder höher zur Verfügung gestellt werden, da andernfalls MQRC_WRONG_GMO_VERSION zurückgegeben wird.

Wenn das im Feld *MsgToken* enthaltene Nachrichtentoken zu diesem Zeitpunkt nicht gültig ist, wird der Code MQCC_FAILED mit der Ursache MQRC_NO_MSG_AVAILABLE zurückgegeben, sofern kein anderer Fehler vorliegt.

Optionen (MQLONG)

Die **MQGMO**-Optionen steuern die Aktion von MQGET. Sie können keine oder aber auch mehrere der Optionen angeben. Wenn Sie mehrere optionale Werte benötigen, gehen Sie folgendermaßen vor:

- Fügen Sie die Werte hinzu (fügen Sie dieselbe Konstante nicht mehrmals hinzu) oder
- kombinieren Sie die Werte mithilfe der bitweisen ODER-Operation (sofern die Programmiersprache Bitoperationen unterstützt).

Auf ungültige Kombinationen von Optionen wird hingewiesen, alle anderen Kombinationen sind gültig.

Warteoptionen: Die folgenden Optionen beziehen sich auf das Warten auf Nachrichten und deren Eintreffen in der Warteschlange:

MQGMO_WAIT
Die Anwendung wartet, bis eine geeignete Nachricht eintrifft. Im Feld *WaitInterval* ist angegeben, wie lange die Anwendung höchstens wartet.

Wichtig: Ist sofort eine angemessene Nachricht verfügbar, kommt es zu keiner Wartezeit oder Verzögerung.

Wenn MQGET-Anforderungen unterdrückt werden oder MQGET-Anforderungen während des Wartens unterdrückt werden, wird das Warten abgebrochen. Selbst wenn die Warteschlange geeignete Nachrichten enthält, wird der Aufruf mit dem Code MQCC_FAILED und dem Ursachencode MQRC_GET_INHIBITED beendet.

Sie können MQGMO_WAIT in Verbindung mit der Option MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT verwenden.

Wenn mehrere Anwendungen in derselben gemeinsam genutzten Warteschlange warten, wird mit folgenden Regeln ausgewählt, welche Anwendung beim Eintreffen einer geeigneten Nachricht aktiviert wird:

Tabelle 505. Regeln für die Aktivierung von MQGET-Aufrufen in einer gemeinsam genutzten Warteschlange		
Anzahl der MQGET-Aufrufe, die auf ihre Aktivierung warten		**Ergebnis**
Mit BROWSE-Option	**Ohne DURCHSUCHEN-Option[1]**	
--	Mindestens einer	Ein MQGET-Aufruf ohne BROWSE-Option wird aktiviert.
Mindestens einer	--	Alle MQGET-Aufrufe mit einer BROWSE-Option werden aktiviert.
Mindestens einer	Mindestens einer	Ein MQGET-Aufruf ohne BROWSE-Option wird aktiviert. Die Anzahl der aktivierten MQGET-Aufrufe mit einer BROWSE-Option ist unvorhersehbar.

Wenn mehrere MQGET-Aufrufe ohne BROWSE-Option in derselben Warteschlange warten, wird nur ein Aufruf aktiviert. Der Warteschlangenmanager versucht, die wartenden Aufrufe in der folgenden Prioritätsfolge zu berücksichtigen:

1. Spezifische Abrufanforderungen mit Warteoption (get-wait), die nur von bestimmten Nachrichten erfüllt werden können, beispielsweise von Nachrichten mit einem bestimmten Wert im Feld *MsgId* und/oder *CorrelId*.
2. Allgemeine Abrufanforderungen mit Warteoption (get-wait), die von jeder Nachricht erfüllt werden können.

Anmerkung:

- Innerhalb der ersten Kategorie wird spezifischeren Abrufanforderungen mit Warteoption keine zusätzliche Priorität zugewiesen. Dies gilt beispielsweise für Anforderungen, in denen sowohl ein Wert für die Nachrichten-ID (*MsgId*) als auch die Korrelations-ID (*CorrelId*) angegeben ist.
- Bei beiden Kategorien kann nicht vorhergesagt werden, welche Anwendung ausgewählt wird. Insbesondere muss beachtet werden, dass nicht unbedingt die Anwendung ausgewählt wird, die am längsten wartet.
- Die Pfadlänge und Vorrangsteuerungsaspekte des Betriebssystems können dazu führen, dass eine wartende Anwendung mit einer niedrigeren Betriebssystempriorität als erwartet die Nachricht abruft.
- Es kann außerdem vorkommen, dass eine Anwendung, die nicht wartet, die Nachricht anstelle einer wartenden Anwendung abruft.

Unter z/OS gelten die folgenden Punkte:

- Wenn die Anwendung während des Wartens auf eine einzutreffende Nachricht mit der sonstigen Verarbeitung fortfahren soll, kann es sinnvoll sein, stattdessen die Signaloption (MQGMO_SET_SIGNAL) zu verwenden. Die Signaloption ist jedoch umgebungsspezifisch; sie darf nicht von Anwendungen verwendet werden, die Sie zwischen verschiedenen Umgebungen portieren.
- Falls mehrere MQGET-Aufrufe, die sowohl Warte- als auch Signaloptionen verwenden, auf dieselbe Nachricht warten, wird jeder wartende Aufruf gleichermaßen berücksichtigt. Die Angabe von MQGMO_SET_SIGNAL zusammen mit MQGMO_WAIT gilt als Fehler. Die Angabe dieser Option in Verbindung mit einer Warteschlangenkennung, für die ein Signal aussteht, gilt ebenfalls als Fehler.
- Wenn Sie MQGMO_WAIT oder MQGMO_SET_SIGNAL für eine Warteschlange angeben, deren *IndexType*-Feld den Wert MQIT_MSG_TOKEN enthält, sind keine Auswahlkriterien zulässig. Dies bedeutet Folgendes:
  - Wenn Sie eine MQGMO-Struktur der Version 1 verwenden, müssen Sie in dem MQMD, der im Aufruf MQGET angegeben ist, die Felder *MsgId* und *CorrelId* auf MQMI_NONE und MQCI_NONE setzen.

[1] Ein MQGET -Aufruf, der die Option MQGMO_LOCK angibt, wird als Aufruf ohne Anzeige behandelt.

– Wenn Sie eine MQGMO-Struktur der Version 2 oder höher verwenden, müssen Sie das Feld *MatchOptions* auf MQMO_NONE setzen.

- Wenn bei einem MQGET-Aufruf für eine gemeinsam genutzte Warteschlange der Aufruf eine Anzeigeaufforderung oder ein Abruf einer Gruppennachricht mit Löschen ist und wenn weder *MsgId* noch *CorrelId* abgeglichen werden sollen, wird der MQGET-Aufruf alle 200 Millisekunden erneut abgesetzt, bis in der Warteschlange eine passende Nachricht ankommt oder das Warteintervall abläuft.

  Diese Methode verursacht einen nicht erwarteten Verarbeitungsaufwand und ist kein effizientes Verfahren zum Abrufen von Nachrichten, wenn nur selten neue Nachrichten eintreffen. Im Fall einer Anzeigeaufforderung können Sie diesem Systemaufwand vorbeugen, indem Sie im MQGET-Aufruf den Abgleich mit *MsgId* (falls nicht indiziert oder durch *MsgId* indiziert) oder mit *CorrelId* (falls durch *CorrelId* indiziert) festlegen.

Die Option MQGMO_WAIT wird ignoriert, wenn sie zusammen mit MQGMO_BROWSE_MSG_UNDER_CURSOR oder MQGMO_MSG_UNDER_CURSOR angegeben wird. In diesem Fall wird kein Fehler ausgegeben.

MQGMO_NO_WAIT
Die Anwendung wartet nicht, wenn keine geeignete Nachricht verfügbar ist. MQGMO_NO_WAIT ist das Gegenteil von MQGMO_WAIT. MQGMO_NO_WAIT wird zur Unterstützung der Programmdokumentation definiert. Es handelt sich dabei um den Standardwert, wenn nichts anderes angegeben ist.

MQGMO_SET_SIGNAL
Verwenden Sie diese Option in Verbindung mit den Feldern *Signal1* und *Signal2*. Sie ermöglicht Anwendungen, ihre sonstige Verarbeitung fortzusetzen, solange sie auf das Eintreffen einer Nachricht warten. Darüber hinaus können Anwendungen mit dieser Option auf das Eintreffen von Nachrichten in mehreren Warteschlangen warten (sofern geeignete Betriebssystemfunktionen verfügbar sind).

Anmerkung: Da die Option MQGMO_SET_SIGNAL umgebungsspezifisch ist, darf sie nicht für Anwendungen verwendet werden, die portiert werden sollen.

In zwei Situationen wird der Aufruf so beendet, als ob diese Option nicht angegeben worden wäre:

1. Wenn eine derzeit verfügbare Nachricht die im Nachrichtendeskriptor angegebenen Kriterien erfüllt.
2. Wenn ein Parameterfehler oder ein sonstiger synchroner Fehler festgestellt wird.

Wenn derzeit keine Nachricht verfügbar ist, die die im Nachrichtendeskriptor angegebenen Kriterien erfüllt, wird die Steuerung an die Anwendung zurückgegeben, ohne dass auf das Eintreffen einer Nachricht gewartet wird. Die Parameter *CompCode* und *Reason* sind auf MQCC_WARNING und MQRC_SIGNAL_REQUEST_ACCEPTED gesetzt. Andere Ausgabefelder im Nachrichtendeskriptor und die Ausgabeparameter des Aufrufs MQGET werden nicht festgelegt. Wenn zu einem späteren Zeitpunkt eine geeignete Nachricht eintrifft, wird das Signal durch die Übergabe des Ereignissteuerblocks übermittelt.

Das aufrufende Modul muss dann den Aufruf MQGET erneut ausgeben, um die Nachricht abzurufen. Die Anwendung kann unter Verwendung der vom Betriebssystem bereitgestellten Funktionen auf dieses Signal warten.

Wenn das Betriebssystem einen mehrfachen Wartemechanismus bereitstellt, können Sie damit in einer beliebigen Warteschlange unter mehreren Warteschlangen auf das Eintreffen einer Nachricht warten.

Wird im Feld *WaitInterval* ein Warteintervall ungleich null angegeben, wird das Signal nach Ablauf des Warteintervalls übermittelt. Der Warteschlangenmanager kann den Wartevorgang auch abbrechen. In diesem Fall wird das Signal übermittelt.

Mehrere MQGET-Aufrufe können ein Signal für dieselbe Nachricht festlegen. Die Aktivierungsreihenfolge der Anwendungen entspricht derjenigen, die für die Option MQGMO_WAIT beschrieben wird.

Falls mehrere MQGET-Aufrufe auf dieselbe Nachricht warten, wird jeder wartende Aufruf gleichermaßen berücksichtigt. Die Aufrufe können eine Kombination aus Warte- und Signaloptionen enthalten.

Unter bestimmten Bedingungen kann der MQGET-Aufruf eine Nachricht abrufen, und ein Signal, das aus dem Eintreffen derselben Nachricht resultiert, kann übermittelt werden. Wird ein Signal übermittelt, muss eine Anwendung darauf vorbereitet sein, dass keine Nachricht verfügbar ist.

Eine Warteschlangenkennung darf nur über eine ausstehende Signalanforderung verfügen.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- MQGMO_UNLOCK
- MQGMO_WAIT

Wenn bei einem MQGET-Aufruf für eine gemeinsam genutzte Warteschlange der Aufruf eine Anzeigeanforderung oder ein Abruf einer Gruppennachricht mit Löschen ist und wenn weder *MsgId* noch *CorrelId* abgeglichen werden sollen, wird nach 200 Millisekunden MQEC_MSG_ARRIVED an den Signal-Ereignissteuerblock des Benutzers gesendet.

Dies geschieht auch dann, wenn noch keine passende Nachricht in der Warteschlange eingetroffen ist, bis das Warteintervall abgelaufen ist, wenn an die Warteschlange MQEC_WAIT_INTERVAL_EXPIRED gesendet wird. Wenn MQEC_MSG_ARRIVED gesendet wird, müssen Sie einen zweiten MQGET-Aufruf ausgeben, um die Nachricht (sofern verfügbar) abzurufen.

Dieses Verfahren soll sicherstellen, dass Sie zeitnah über den Eingang einer Nachricht informiert werden; im Vergleich zu einer ähnlichen Aufrufsequenz in einer nicht gemeinsam genutzten Warteschlange kann es jedoch als nicht erwarteter Verarbeitungsaufwand erscheinen.

Diese Methode ist kein effizientes Verfahren zum Abrufen von Nachrichten, wenn nur selten neue Nachrichten eintreffen. Um diesen Aufwand für den Suchfall zu vermeiden, geben Sie *MsgId* (wenn nicht indexiert oder durch *MsgId*indexiert) oder *CorrelId* (wenn indexiert durch *CorrelId*) im Aufruf MQGET an.

Diese Option wird nur unter z/OS unterstützt.

MQGMO_FAIL_IF_QUIESCING
Das Fehlschlagen des MQGET-Aufrufs wird erzwungen, wenn sich der Warteschlangenmanager im Stilllegungsstatus befindet.

Unter z/OS erzwingt diese Option auch dann ein Fehlschlagen des MQGET-Aufrufs, wenn sich die Verbindung (bei einer CICS- oder IMS-Anwendung) im Stilllegungsstatus befindet.

Wird diese Option zusammen mit MQGMO_WAIT oder MQGMO_SET_SIGNAL angegeben und steht zum Zeitpunkt des Warteschlangenmanagerübergangs in den Stilllegungsstatus ein Wartevorgang oder Signal aus, tritt Folgendes ein:

- Der Wartevorgang wird abgebrochen und der Aufruf gibt den Beendigungscode MQCC_FAILED mit dem Ursachencode MQRC_Q_MGR_QUIESCING oder MQRC_CONNECTION_QUIESCING zurück.
- Das Signal wird mit einem umgebungsspezifischen Signalbeendigungscode abgebrochen.

  Unter z/OS wird das Signal mit dem Ereignisbeendigungscode MQEC_Q_MGR_QUIESCING oder MQEC_CONNECTION_QUIESCING beendet.

Wenn MQGMO_FAIL_IF_QUIESCING nicht angegeben ist und der Warteschlangenmanager oder die Verbindung in den Stilllegungsstatus übergeht, werden weder der Wartevorgang noch die Signalausgabe abgebrochen.

Synchronisationspunktoptionen: Die folgenden Optionen beziehen sich auf die Verwendung des MQGET-Aufrufs in einer Arbeitseinheit:

MQGMO_SYNCPOINT
Die Anforderung wird innerhalb der normalen Arbeitseinheitenprotokolle ausgeführt. Die Nachricht ist als nicht verfügbar für andere Anwendungen markiert, wird jedoch nur dann aus der Warteschlange gelöscht, wenn die Arbeitseinheit festgeschrieben wird. Die Nachricht steht wieder zur Verfügung, wenn die Arbeitseinheit zurückgesetzt wird.

Die Optionen MQGMO_SYNCPOINT und MQGMO_NO_SYNCPOINT müssen nicht festgelegt werden. In diesem Fall wird die Einbeziehung der Abrufanforderung in Arbeitseinheitenprotokolle durch die Um-

gebung bestimmt, in der der Warteschlangenmanager aktiv ist. Sie wird nicht durch die Umgebung bestimmt, in der die Anwendung ausgeführt wird. Unter z/OS befindet sich die Abrufanforderung innerhalb einer Arbeitseinheit. In allen anderen Umgebungen befindet sich die Abrufanforderung nicht innerhalb einer Arbeitseinheit.

Aufgrund dieses Unterschieds darf eine Anwendung, die Sie portieren möchten, diese Option nicht als Standardeinstellung zulassen; geben Sie explizit MQGMO_SYNCPOINT oder MQGMO_NO_SYNCPOINT an.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- 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
Die Anforderung soll innerhalb der normalen Arbeitseinheitenprotokolle ausgeführt werden, allerdings *nur*, wenn die abgerufene Nachricht persistent ist. Eine persistente Nachricht hat den Wert MQPER_PERSISTENT im Feld *Persistence* im MQMD.

- Wenn die Nachricht persistent ist, verarbeitet der Warteschlangenmanager den Aufruf, als ob die Anwendung MQGMO_SYNCPOINT angegeben hätte.
- Wenn die Nachricht nicht persistent ist, verarbeitet der Warteschlangenmanager den Aufruf, als ob die Anwendung MQGMO_NO_SYNCPOINT angegeben hätte.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- 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

Diese Option wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, z/OS, IBM i, Solaris und Linuxsowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

MQGMO_NO_SYNCPOINT
Die Anforderung soll außerhalb der normalen Arbeitseinheitenprotokolle ausgeführt werden. Wenn Sie eine Nachricht ohne Suchoption abrufen, wird sie unverzüglich aus der Warteschlange gelöscht. Die Nachricht kann nicht durch das Zurücksetzen der Arbeitseinheit erneut verfügbar gemacht werden.

Diese Option wird vorausgesetzt, wenn Sie MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT angeben.

Die Optionen MQGMO_SYNCPOINT und MQGMO_NO_SYNCPOINT müssen nicht festgelegt werden. In diesem Fall wird die Einbeziehung der Abrufanforderung in Arbeitseinheitenprotokolle durch die Umgebung bestimmt, in der der Warteschlangenmanager aktiv ist. Sie wird nicht durch die Umgebung bestimmt, in der die Anwendung ausgeführt wird. Unter z/OS befindet sich die Abrufanforderung innerhalb einer Arbeitseinheit. In allen anderen Umgebungen befindet sich die Abrufanforderung nicht innerhalb einer Arbeitseinheit.

Aufgrund dieses Unterschieds darf eine Anwendung, die Sie portieren möchten, diese Option nicht als Standardeinstellung zulassen; geben Sie explizit entweder MQGMO_SYNCPOINT oder MQGMO_NO_SYNCPOINT an.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT
Eine Arbeitseinheit wird zurückgesetzt, ohne die Nachricht, die mit dieser Option markiert wurde, in der Warteschlange wiederherzustellen.

Diese Option wird nur unter z/OS unterstützt.

Wenn diese Option angegeben wird, muss MQGMO_SYNCPOINT ebenfalls angegeben werden. MQGMO_MARK_SKIP_BACKOUT ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Anmerkung: In IMS und CICS müssen Sie möglicherweise einen zusätzlichen WebSphere MQ-Aufruf ausgeben, nachdem Sie eine Arbeitseinheit zurückgesetzt haben, die eine mit MQGMO_MARK_SKIP_BACKOUT markierte Nachricht enthält. Sie müssen einen WebSphere-MQ-Aufruf ausgeben, bevor Sie die neue Arbeitseinheit festschreiben, die die markierte Nachricht enthält. Dieser Aufruf kann ein beliebiger WebSphere MQ-Aufruf sein.

1. In IMS, wenn Sie den IMS-APAR PN60855 nicht angewendet haben und eine IMS-MPP- oder -BMP-Anwendung ausführen.
2. In CICS, wenn Sie eine Anwendung ausführen.

Geben Sie in beiden Fällen einen beliebigen WebSphere MQ-Aufruf aus, bevor Sie die neue Arbeitseinheit mit der zurückgesetzten Nachricht festschreiben.

Anmerkung: Innerhalb einer Arbeitseinheit darf nur eine Abrufanforderung für das Überspringen der Zurücksetzung markiert sein, während keine oder mehrere nicht markierte Abrufanforderungen vorhanden sein können.

Wenn eine Anwendung eine Arbeitseinheit zurücksetzt, wird für eine Nachricht, die mit MQGMO_MARK_SKIP_BACKOUT abgerufen wurde, nicht ihr vorheriger Zustand wiederhergestellt. Andere Ressourcenaktualisierungen werden zurückgesetzt. Die Nachricht wird behandelt, als ob sie in einer neuen Arbeitseinheit abgerufen worden wäre, die durch die Zurücksetzungsanforderung gestartet wurde. Die Nachricht wird ohne die Option MQGMO_MARK_SKIP_BACKOUT abgerufen.

MQGMO_MARK_SKIP_BACKOUT ist nützlich, wenn nach der Änderung einiger Ressourcen offensichtlich ist, dass die Arbeitseinheit nicht erfolgreich abgeschlossen werden kann. Wenn Sie diese Option übergehen, wird die Nachricht durch das Zurücksetzen der Arbeitseinheit in der Warteschlange wiederhergestellt. Dieselbe Ereignisfolge tritt auf, wenn die Nachricht beim nächsten Mal abgerufen wird.

Wenn Sie MQGMO_MARK_SKIP_BACKOUT jedoch im ursprünglichen MQGET-Aufruf angeben, werden die Aktualisierungen der anderen Ressourcen durch das Zurücksetzen der Arbeitseinheit zurückgesetzt. Die Nachricht wird behandelt, als ob sie in einer neuen Arbeitseinheit abgerufen worden wäre. Die Anwendung kann eine geeignete Fehlerbehandlung ausführen. Sie kann eine Berichtsnachricht an den Absender der ursprünglichen Nachricht senden oder die ursprüngliche Nachricht in die Warteschlange für nicht zustellbare Nachrichten stellen. Anschließend kann sie die neue Arbeitseinheit

festschreiben. Durch die Festschreibung der neuen Arbeitseinheit wird die Nachricht dauerhaft aus der ursprünglichen Warteschlange entfernt.

MQGMO_MARK_SKIP_BACKOUT markiert eine einzelne physische Nachricht. Wenn die Nachricht zu einer Nachrichtengruppe gehört, werden die übrigen Nachrichten in der Gruppe nicht markiert. Ebenso werden die übrigen Segmente in der logischen Nachricht nicht markiert, wenn die markierte Nachricht ein Segment einer logischen Nachricht ist.

Jede Nachricht in einer Gruppe kann markiert werden, aber wenn Nachrichten unter Verwendung von MQGMO_LOGICAL_ORDER abgerufen werden, ist die Markierung der ersten Nachricht in der Gruppe von Vorteil. Falls die Arbeitseinheit zurückgesetzt wird, wird die erste (markierte) Nachricht in die neue Arbeitseinheit verschoben. Die zweite Nachricht sowie nachfolgende Nachrichten in der Gruppe werden in der Warteschlange wiederhergestellt. Die in der Warteschlange verbleibenden Nachrichten können nicht unter Verwendung von MQGMO_LOGICAL_ORDER durch eine andere Anwendung abgerufen werden. Die erste Nachricht in der Gruppe befindet sich nicht mehr in der Warteschlange. Allerdings kann die Anwendung, die die Arbeitseinheit zurückgesetzt hat, die zweite und nachfolgende Nachrichten unter Verwendung der Option MQGMO_LOGICAL_ORDER in die neue Arbeitseinheit abrufen. Die erste Nachricht wurde bereits abgerufen.

Gelegentlich kann es vorkommen, dass Sie die neue Arbeitseinheit zurücksetzen müssen. Dies kann beispielsweise der Fall sein, wenn die Warteschlange für nicht zustellbare Nachrichten voll ist und die Nachricht nicht gelöscht werden darf. Durch die Zurücksetzung der neuen Arbeitseinheit wird die Nachricht in der ursprünglichen Warteschlange wiederhergestellt, was den Verlust der Nachricht verhindert. In dieser Situation kann die Verarbeitung jedoch nicht fortgesetzt werden. Nach der Zurücksetzung der neuen Arbeitseinheit muss die Anwendung den Operator oder Administrator informieren, dass ein nicht behebbarer Fehler vorliegt. Anschließend wird sie beendet.

MQGMO_MARK_SKIP_BACKOUT funktioniert nur, wenn die Arbeitseinheit, die die Abrufanforderung enthält, von der Anwendung unterbrochen wird, die sie zurücksetzt. Wenn die Arbeitseinheit mit der Abrufanforderung zurückgesetzt wird, weil die Transaktion fehlschlägt oder das System ausfällt, wird MQGMO_MARK_SKIP_BACKOUT ignoriert. Jede Nachricht, die mit dieser Option abgerufen wird, wird auf dieselbe Weise in der Warteschlange wiederhergestellt wie Nachrichten, die ohne diese Option abgerufen werden.

Anzeigeoptionen: Die folgenden Optionen beziehen sich auf das Anzeigen der Nachrichten in der Warteschlange:

MQGMO_BROWSE_FIRST
Wenn eine Warteschlange mit der Option MQOO_BROWSE geöffnet wird, wird ein Anzeigecursor eingerichtet, der logisch vor der ersten Nachricht in der Warteschlange positioniert wird. Sie können dann MQGET-Aufrufe mit der Option MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT oder MQGMO_BROWSE_MSG_UNDER_CURSOR verwenden, um Nachrichten aus der Warteschlange abzurufen, ohne dass diese dabei gelöscht werden. Der Anzeigecursor markiert innerhalb der Nachrichten in der Warteschlange die Position, ab der der nächste MQGET-Aufruf mit MQGMO_BROWSE_NEXT nach einer geeigneten Nachricht sucht.

MQGMO_BROWSE_FIRST ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Es tritt auch ein Fehler auf, wenn die Warteschlange nicht zur Anzeige geöffnet war.

Ein MQGET-Aufruf mit der Option MQGMO_BROWSE_FIRST ignoriert die vorherige Position des Anzeigecursors. Die erste Nachricht in der Warteschlange, die die im Nachrichtendeskriptor angegebenen

Bedingungen erfüllt, wird abgerufen. Die Nachricht verbleibt in der Warteschlange und der Anzeigecursor wird auf der Nachricht platziert.

Nach diesem Aufruf ist der Anzeigecursor auf der zurückgegebenen Nachricht positioniert. Es kann vorkommen, dass die Nachricht vor Ausgabe des nächsten MQGET-Aufrufs mit der Option MQGMO_BROWSE_NEXT aus der Warteschlange entfernt wird. Obwohl die betreffende Position jetzt leer ist, bleibt in diesem Fall der Anzeigecursor an der Position in der Warteschlange, an der sich zuvor die Nachricht befand.

Bei Verwendung der Option MQGMO_MSG_UNDER_CURSOR für einen MQGET-Aufruf ohne Suchoption wird die Nachricht aus der Warteschlange entfernt.

Der Anzeigecursor wird auch dann nicht von einem MQGET-Aufruf ohne Suchoption verschoben, wenn dieselbe *Hobj*-Kennung verwendet wird. Er wird außerdem auch nicht von einem MQGET-Aufruf mit Suchoption verschoben, der den Beendigungscode MQCC_FAILED oder den Ursachencode MQRC_TRUNCATED_MSG_FAILED zurückgibt.

Geben Sie zusammen mit dieser Option die Option MQGMO_LOCK an, damit die durchsuchte Nachricht gesperrt wird.

Sie können MQGMO_BROWSE_FIRST mit jeder gültigen Kombination der Optionen des Typs MQGMO_* und MQMO_* angeben, die die Verarbeitung von Nachrichten in Gruppen und von Segmenten logischer Nachrichten steuern.

Wenn Sie MQGMO_LOGICAL_ORDER angeben, werden die Nachrichten in logischer Reihenfolge durchsucht. Falls Sie diese Option übergehen, erfolgt das Browsing der Nachrichten in physischer Reihenfolge. Wenn Sie MQGMO_BROWSE_FIRST angeben, können Sie zwischen der logischen und physischen Reihenfolge wechseln. Nachfolgende MQGET-Aufrufe, die die Option MQGMO_BROWSE_NEXT verwenden, durchsuchen die Warteschlange in derselben Reihenfolge wie der zuletzt ausgeführte Aufruf, bei dem MQGMO_BROWSE_FIRST für die Warteschlangenkennung angegeben war.

Der Warteschlangenmanager behält zwei Mengen mit Gruppen- und Segmentinformationen für MQGET-Aufrufe bei. Die Gruppen- und Segmentinformationen für Aufrufe mit Suchoption werden gesondert von den Informationen für Aufrufe gespeichert, bei denen Nachrichten aus der Warteschlange entfernt werden. Wenn Sie MQGMO_BROWSE_FIRST angeben, ignoriert der Warteschlangenmanager die Gruppen- und Segmentinformationen beim Browsing. Er durchsucht die Warteschlange, als ob keine aktuelle Gruppe und keine aktuelle logische Nachricht vorhanden wären. Verläuft der MQGET-Aufruf mit dem Ergebnis MQCC_OK oder MQCC_WARNING erfolgreich, werden die Gruppen- und Segmentinformationen für das Browsing auf die Werte der zurückgegebenen Nachricht gesetzt. Wenn der Aufruf fehlschlägt, bleiben die vor dem Aufruf vorhandenen Gruppen- und Segmentinformationen unverändert erhalten.

MQGMO_BROWSE_NEXT
Der Anzeigecursor rückt zur nächsten Nachricht in der Warteschlange vor, die die Auswahlkriterien erfüllt, welche im Aufruf MQGET angegeben sind. Die Nachricht wird an die Anwendung zurückgegeben, verbleibt jedoch in der Warteschlange.

MQGMO_BROWSE_NEXT ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Es tritt auch ein Fehler auf, wenn die Warteschlange nicht zur Anzeige geöffnet war.

MQGMO_BROWSE_NEXT verhält sich genau wie MQGMO_BROWSE_FIRST, wenn es sich um den ersten Aufruf zum Durchsuchen einer Warteschlange handelt, seit diese für das Browsing geöffnet wurde.

Es kann vorkommen, dass die Nachricht unter dem Cursor vor Ausgabe des nächsten MQGET-Aufrufs mit der Option MQGMO_BROWSE_NEXT aus der Warteschlange entfernt wird. Obwohl die betreffende Position jetzt leer ist, bleibt der Anzeigecursor logisch an der Position in der Warteschlange, an der sich zuvor die Nachricht befand.

Nachrichten werden auf eine von zwei möglichen Arten in der Warteschlange gespeichert:

- First In/First Out unter Berücksichtigung der Priorität (MQMDS_PRIORITY) oder
- First In/First Out *ungeachtet* der Priorität (MQMDS_FIFO)

Das Warteschlangenattribut *MsgDeliverySequence* gibt an, welche Methode Anwendung findet (der Abschnitt „Attribute für Warteschlangen" auf Seite 833 enthält ausführliche Informationen hierzu).

Angenommen, eine Warteschlange enthält im Feld *MsgDeliverySequence* den Wert MQMDS_PRIORITY. Nun trifft eine Nachricht in der Warteschlange ein, die eine höhere Priorität als die Nachricht hat, auf die derzeit durch den Anzeigecursor gezeigt wird. In diesem Fall wird die Nachricht mit der höheren Priorität während des aktuellen Scanvorgangs der Warteschlange mit MQGMO_BROWSE_NEXT nicht gefunden. Sie kann erst gefunden werden, nachdem der Anzeigecursor mit MQGMO_BROWSE_FIRST zurückgesetzt oder die Warteschlange erneut geöffnet wurde.

Falls erforderlich, kann die Option MQGMO_MSG_UNDER_CURSOR bei einem MQGET-Aufruf ohne Suchfunktion verwendet werden, um die Nachricht aus der Warteschlange zu entfernen.

Der Anzeigecursor wird nicht von MQGET-Aufrufen ohne Suchoption verschoben, wenn dieselbe *Hobj*-Kennung verwendet wird.

Geben Sie zusammen mit dieser Option die Option MQGMO_LOCK an, damit die durchsuchte Nachricht gesperrt wird.

Sie können MQGMO_BROWSE_NEXT mit jeder gültigen Kombination der Optionen des Typs MQGMO_* und MQMO_* angeben, die die Verarbeitung von Nachrichten in Gruppen und von Segmenten logischer Nachrichten steuern.

Wenn Sie MQGMO_LOGICAL_ORDER angeben, werden die Nachrichten in logischer Reihenfolge durchsucht. Falls Sie diese Option übergehen, erfolgt das Browsing der Nachrichten in physischer Reihenfolge. Wenn Sie MQGMO_BROWSE_FIRST angeben, können Sie zwischen der logischen und physischen Reihenfolge wechseln. Nachfolgende MQGET-Aufrufe, die die Option MQGMO_BROWSE_NEXT verwenden, durchsuchen die Warteschlange in derselben Reihenfolge wie der zuletzt ausgeführte Aufruf, bei dem MQGMO_BROWSE_FIRST für die Warteschlangenkennung angegeben war. Der Aufruf schlägt mit dem Ursachencode MQRC_INCONSISTENT_BROWSE fehl, wenn diese Bedingung nicht erfüllt wird.

Anmerkung: Wenn MQGMO_LOGICAL_ORDER nicht angegeben ist, seien Sie bei der Verwendung eines MQGET-Aufrufs für das Browsing über das Ende einer Nachrichtengruppe hinaus vorsichtig. Nehmen Sie beispielsweise an, dass die letzte Nachricht in der Gruppe vor der ersten Nachricht in der Gruppe in der Warteschlange steht. Wenn Sie MQGMO_BROWSE_NEXT verwenden, um über das Ende der Gruppe hinaus zu blättern, wird bei Angabe von MQMO_MATCH_MSG_SEQ_NUMBER mit *MsgSeqNumber*, das auf 1 gesetzt wurde, die erste Nachricht in der bereits durchsuchten Gruppe zurückgegeben. Dies kann sofort eintreten oder mehrere MQGET-Aufrufe später, wenn Zwischengruppen verfügbar sind.Die gleiche Überlegung gilt für eine logische Nachricht, die nicht in einer Gruppe enthalten ist.

Die Gruppen- und Segmentinformationen für Aufrufe mit Suchoption werden gesondert von den Informationen für Aufrufe gespeichert, bei denen Nachrichten aus der Warteschlange entfernt werden.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Die Nachricht, auf die der Anzeigecursor zeigt, wird ohne anschließendes Löschen der Nachricht abgerufen, und zwar unabhängig von den Optionen des Typs MQMO_*, die im Feld *MatchOptions* in MQGMO angegeben sind.

MQGMO_BROWSE_MSG_UNDER_CURSOR ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT

- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Es tritt auch ein Fehler auf, wenn die Warteschlange nicht zur Anzeige geöffnet war.

Die Nachricht, auf die der Anzeigecursor zeigt, ist die zuletzt abgerufene Nachricht, bei der die Option MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT verwendet wurde. Der Aufruf schlägt fehl, wenn für diese Warteschlange seit ihrer Öffnung keiner dieser Aufrufe ausgegeben wurde. Der Aufruf schlägt ebenfalls fehl, wenn die Nachricht unter dem Anzeigecursor seither abgerufen und anschließend gelöscht wurde.

Die Position des Anzeigecursors wird durch diesen Aufruf nicht geändert.

Die Option MQGMO_MSG_UNDER_CURSOR kann bei einem MQGET-Aufruf ohne Suchfunktion verwendet werden, um die Nachricht aus der Warteschlange zu entfernen.

Der Anzeigecursor wird auch dann nicht von einem MQGET-Aufruf ohne Suchoption verschoben, wenn dieselbe *Hobj*-Kennung verwendet wird. Er wird außerdem auch nicht von einem MQGET-Aufruf mit Suchoption verschoben, der den Beendigungscode MQCC_FAILED oder den Ursachencode MQRC_TRUNCATED_MSG_FAILED zurückgibt.

Wenn MQGMO_BROWSE_MSG_UNDER_CURSOR mit MQGMO_LOCK angegeben wird, gilt Folgendes:

- Falls bereits eine Nachricht gesperrt ist, muss sie sich unter dem Cursor befinden, damit sie ohne Entsperren und erneutes Sperren zurückgegeben wird. Die Nachricht bleibt gesperrt.
- Wenn keine gesperrte Nachricht vorhanden ist und sich eine Nachricht unter dem Anzeigecursor befindet, wird sie gesperrt und an die Anwendung zurückgegeben. Befindet sich keine Nachricht unter dem Anzeigecursor, schlägt der Aufruf fehl.

Wenn MQGMO_BROWSE_MSG_UNDER_CURSOR ohne MQGMO_LOCK angegeben wird, gilt Folgendes:

- Wenn bereits eine Nachricht gesperrt ist, muss es sich dabei um die Nachricht unter dem Anzeigecursor handeln. Die Nachricht wird an die Anwendung zurückgegeben und anschließend entsperrt. Da die Nachricht jetzt entsperrt ist, kann nicht garantiert werden, dass sie von derselben Anwendung erneut durchsucht oder abgerufen und anschließend gelöscht werden kann. Möglicherweise wurde sie von einer anderen Anwendung, die Nachrichten aus der Warteschlange abruft, abgerufen und anschließend gelöscht.
- Wenn keine gesperrte Nachricht vorhanden ist und sich eine Nachricht unter dem Anzeigecursor befindet, wird sie an die Anwendung zurückgegeben. Befindet sich keine Nachricht unter dem Anzeigecursor, schlägt der Aufruf fehl.

Wenn MQGMO_COMPLETE_MSG mit MQGMO_BROWSE_MSG_UNDER_CURSOR angegeben wird, muss der Anzeigecursor eine Nachricht angeben, bei der das Feld *Offset* im MQMD den Wert null enthält. Wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_INVALID_MSG_UNDER_CURSOR fehl.

Die Gruppen- und Segmentinformationen für Aufrufe mit Suchoption werden gesondert von den Informationen für Aufrufe gespeichert, bei denen Nachrichten aus der Warteschlange entfernt werden.

MQGMO_MSG_UNDER_CURSOR
Die Nachricht, auf die der Anzeigecursor zeigt, wird abgerufen, und zwar unabhängig von den Optionen des Typs MQMO_*, die im Feld *MatchOptions* in MQGMO angegeben sind. Dabei wird die Nachricht aus der Warteschlange entfernt.

Die Nachricht, auf die der Anzeigecursor zeigt, ist die zuletzt abgerufene Nachricht, bei der die Option MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT verwendet wurde.

Wenn MQGMO_COMPLETE_MSG mit MQGMO_MSG_UNDER_CURSOR angegeben wird, muss der Anzeigecursor eine Nachricht angeben, bei der das Feld *Offset* im MQMD den Wert null enthält. Wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_INVALID_MSG_UNDER_CURSOR fehl.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

Es tritt auch ein Fehler auf, wenn die Warteschlange nicht sowohl zur Anzeige als auch zur Eingabe geöffnet war. Wenn der Anzeigecursor derzeit nicht auf eine abrufbare Nachricht zeigt, wird vom MQGET-Aufruf ein Fehler zurückgegeben.

MQGMO_MARK_BROWSE_HANDLE
Die Nachricht, die von einem erfolgreichen MQGET-Aufruf zurückgegeben oder durch das im Feld *MsgToken* zurückgegebene Nachrichtentoken identifiziert wird, wird markiert. Die Markierung gilt ausschließlich für die Objektkennung, die im Aufruf verwendet wird.

Dabei wird die Nachricht nicht aus der Warteschlange entfernt.

MQGMO_MARK_BROWSE_HANDLE ist nur gültig, wenn eine der folgenden Optionen ebenfalls angegeben wird:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Die Nachricht bleibt in diesem Status, bis eines der folgenden Ereignisse eintritt:

- Die betroffene Objektkennung wird auf normale oder sonstige Weise geschlossen.
- Die Markierung der Nachricht wird für diese Kennung durch einen Aufruf von MQGET mit der Option MQGMO_UNMARK_BROWSE_HANDLE aufgehoben.
- Die Nachricht wird von einem zerstörerischen MQGET-Aufruf zurückgegeben, der mit MQCC_OK oder MQCC_WARNING beendet wird. Der Nachrichtenstatus bleibt auch dann geändert, wenn der MQGET-Aufruf zu einem späteren Zeitpunkt zurückgesetzt wird.
- Die Nachricht läuft ab.

MQGMO_MARK_BROWSE_CO_OP
Die Nachricht, die von einem erfolgreichen MQGET-Aufruf zurückgegeben oder durch das im Feld *MsgToken* zurückgegebene Nachrichtentoken identifiziert wird, wird für alle Kennungen in der kooperierenden Gruppe markiert.

Die Markierung auf der kooperativen Ebene erfolgt zusätzlich zu einer eventuell bereits festgelegten Markierung auf Kennungsebene.

Dabei wird die Nachricht nicht aus der Warteschlange entfernt.

MQGMO_MARK_BROWSE_CO_OP ist nur gültig, wenn die verwendete Objektkennung von einem MQOPEN-Aufruf zurückgegeben wurde, bei dem MQOO_CO_OP angegeben war. Sie müssen außerdem eine der folgenden MQGMO-Optionen angeben:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR

- MQGMO_BROWSE_NEXT

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Wenn die Nachricht bereits markiert ist und die Option MQGMO_UNMARKED_BROWSE_MSG nicht angegeben wurde, schlägt der Aufruf mit MQCC_FAILED und dem Ursachencode MQRC_MSG_MARKED_BROWSE_CO_OP fehl.

Die Nachricht bleibt in diesem Status, bis eines der folgenden Ereignisse eintritt:

- Alle Objektkennungen in der kooperierenden Gruppe werden geschlossen.
- Die Markierung der Nachricht wird für kooperierende Browser durch einen Aufruf von MQGET mit der Option MQGMO_UNMARK_BROWSE_CO_OP aufgehoben.
- Die Markierung der Nachricht wird automatisch vom Warteschlangenmanager aufgehoben.
- Die Nachricht wird von einem MQGET-Aufruf ohne Suchoption zurückgegeben. Der Nachrichtenstatus bleibt auch dann geändert, wenn der MQGET-Aufruf zu einem späteren Zeitpunkt zurückgesetzt wird.
- Die Nachricht läuft ab.

MQGMO_UNMARKED_BROWSE_MSG
Ein Aufruf von MQGET, bei dem MQGMO_UNMARKED_BROWSE_MSG angegeben ist, gibt eine Nachricht zurück, die für die zugehörige Kennung als nicht markiert gilt. Es wird keine Nachricht zurückgegeben, wenn die Nachricht für die zugehörige Kennung markiert wurde. Wenn die Warteschlange durch einen Aufruf von MQOPEN mit der Option MQOO_CO_OP geöffnet wurde und die Nachricht von einem Mitglied der kooperierenden Gruppe markiert wurde, wird die Nachricht ebenfalls nicht zurückgegeben.

Diese Option ist mit einer der folgenden Optionen nicht zulässig:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP
Nach einem MQGET-Aufruf mit dieser Option gilt die Nachricht für offene Kennungen in der Gruppe der kooperierenden Kennungen nicht mehr als für die kooperierende Gruppe markiert. Wenn die Nachricht vor diesem Aufruf auf Kennungsebene markiert war, gilt diese Markierung auf Kennungsebene nach wie vor.

Die Option MQGMO_UNMARK_BROWSE_CO_OP ist nur in Verbindung mit einer Kennung gültig, die von einem erfolgreichen MQOPEN-Aufruf mit der Option MQOO_CO_OP zurückgegeben wurde. Der MQGET-Aufruf verläuft auch dann erfolgreich, wenn die Nachricht für die kooperierende Kennungsgruppe nicht als markiert gilt.

MQGMO_UNMARK_BROWSE_CO_OP ist bei einem MQGET-Aufruf ohne Suchoption oder in Verbindung mit einer der folgenden Optionen nicht gültig:

- 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
Nach einem MQGET-Aufruf mit dieser Option gilt die gefundene Nachricht nicht mehr als für diese Kennung markiert.

Der Aufruf verläuft auch dann erfolgreich, wenn die Nachricht nicht für diese Kennung markiert wurde.

Diese Option ist bei einem MQGET-Aufruf ohne Suchoption oder in Verbindung mit einer der folgenden Optionen nicht gültig:

- 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

Sperroptionen: Die folgenden Optionen beziehen sich auf das Sperren von Nachrichten in der Warteschlange:

MQGMO_LOCK
Die durchsuchte Nachricht wird gesperrt und ist somit für andere Kennungen, die für die Warteschlange geöffnet wurden, nicht sichtbar. Diese Option kann nur mit einer der folgenden Optionen angegeben werden:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Für jede Warteschlangenkennung kann immer nur eine Nachricht gesperrt werden. Bei der Nachricht kann es sich um eine logische oder physische Nachricht handeln:

- Wenn Sie MQGMO_COMPLETE_MSG angeben, werden alle Nachrichtensegmente, aus denen die logische Nachricht besteht, für die Warteschlangenkennung gesperrt. Alle Nachrichten müssen in der Warteschlange vorhanden sein und für den Abruf zur Verfügung stehen.
- Wenn Sie MQGMO_COMPLETE_MSG übergehen, wird für die Warteschlangenkennung nur eine einzelne physische Nachricht gesperrt. Ist diese Nachricht zufällig ein Segment einer logischen Nachricht, verhindert das gesperrte Segment, dass andere Anwendungen die logische Nachricht unter Verwendung von MQGMO_COMPLETE_MSG abrufen oder durchsuchen können.

Die gesperrte Nachricht befindet sich immer direkt unter dem Anzeigecursor. Die Nachricht kann von einem späteren MQGET-Aufruf, der die Option MQGMO_MSG_UNDER_CURSOR angibt, aus der Warteschlange entfernt werden. Die Nachricht kann auch von anderen MQGET-Aufrufen entfernt werden, die die Warteschlangenkennung verwenden (beispielsweise von einem Aufruf, der die Nachrichten-ID der gesperrten Nachricht angibt).

Wenn der Aufruf den Beendigungscode MQCC_FAILED oder MQCC_WARNING mit dem Ursachencode MQRC_TRUNCATED_MSG_FAILED zurückgibt, wird keine Nachricht gesperrt.

Entfernt die Anwendung die Nachricht nicht aus der Warteschlange, wird die Sperre mit einer der folgenden Aktionen freigegeben:

- Durch die Ausgabe eines anderen MQGET-Aufrufs für diese Kennung, bei dem entweder MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT angegeben wird. Die Sperre wird freigegeben, wenn der Aufruf mit MQCC_OK oder MQCC_WARNING beendet wird. Wird der Aufruf mit MQCC_FAILED beendet, bleibt die Nachricht weiterhin gesperrt. Hierbei gelten jedoch folgende Ausnahmeregelungen:
  - Die Nachricht wird nicht entsperrt, wenn MQCC_WARNING zusammen mit MQRC_TRUNCATED_MSG_FAILED zurückgegeben wird.
  - Die Nachricht wird entsperrt, wenn MQCC_FAILED zusammen mit MQRC_NO_MSG_AVAILABLE zurückgegeben wird.

  Wenn Sie MQGMO_LOCK ebenfalls angeben, wird die zurückgegebene Nachricht gesperrt. Wenn Sie MQGMO_LOCK übergehen, ist nach dem Aufruf keine gesperrte Nachricht vorhanden.

  Wenn Sie MQGMO_WAIT angeben und keine Nachricht sofort verfügbar ist, wird die ursprüngliche Nachricht vor dem Beginn des Wartevorgangs entsperrt.
- Bei Ausgabe eines anderen MQGET-Aufrufs für diese Kennung, bei dem die Option MQGMO_BROWSE_MSG_UNDER_CURSOR ohne MQGMO_LOCK angegeben ist. Die Sperre wird freigegeben, wenn der Aufruf mit MQCC_OK oder MQCC_WARNING beendet wird. Wird der Aufruf mit MQCC_FAILED beendet, bleibt die Nachricht weiterhin gesperrt. Hierbei gelten jedoch folgende Ausnahmeregelungen:
  - Die Nachricht wird nicht entsperrt, wenn MQCC_WARNING zusammen mit MQRC_TRUNCATED_MSG_FAILED zurückgegeben wird.
- Bei Ausgabe eines anderen MQGET-Aufrufs für diese Kennung mit der Option MQGMO_UNLOCK.
- Bei Ausgabe eines MQCLOSE-Aufrufs unter Verwendung der Kennung. Bei dem MQCLOSE-Aufruf kann es sich um einen impliziten Aufruf handeln, der auf die Beendigung der Anwendung zurückzuführen ist.

Es ist keine spezielle Option MQOPEN erforderlich, um MQGMO_LOCKanzugeben, außer MQOO_BROWSE, die erforderlich ist, um eine zugehörige Suchoption anzugeben.

MQGMO_LOCK ist nicht in Verbindung mit einer der folgenden Optionen gültig:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_LOCK ist nicht möglich, wenn Sie einen IBM WebSphere MQ -Client unter HP Integrity NonStop Server für einen z/OS -Warteschlangenmanager verwenden, wenn er von TMF koordiniert wird.

MQGMO_UNLOCK

Die zu entsperrende Nachricht muss zuvor durch einen MQGET-Aufruf mit der Option MQGMO_LOCK gesperrt worden sein. Falls für diese Kennung keine Nachricht gesperrt wurde, wird der Aufruf mit MQCC_WARNING und MQRC_NO_MSG_LOCKED beendet.

Die Parameter *MsgDesc*, *BufferLength*, *Buffer* und *DataLength* werden nicht geprüft oder geändert, wenn Sie MQGMO_UNLOCK angeben. In *Buffer* wird keine Nachricht zurückgegeben.

Für die Angabe von MQGMO_UNLOCK ist keine besondere Öffnungsoption erforderlich (allerdings ist MQOO_BROWSE erforderlich, damit die Sperranforderung überhaupt ausgegeben wird).

Diese Option kann mit Ausnahme der folgenden Optionen nicht zusammen mit anderen Optionen verwendet werden:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Beide genannten Optionen werden vorausgesetzt, und zwar unabhängig davon, ob sie angegeben wurden.

Optionen für Nachrichtendaten: Die folgenden Optionen beziehen sich auf das Verarbeiten der Daten einer Nachricht, die aus der Warteschlange gelesen wird:

MQGMO_ACCEPT_TRUNCATED_MSG
Falls der Nachrichtenpuffer nicht für die vollständige Nachricht ausreicht, wird mit dieser Option das Füllen des Puffers durch den MQGET- Aufruf zugelassen. MQGET füllt den Puffer mit möglichst viel Nachrichteninhalt. Der Aufruf gibt einen Warnbeendigungscode aus und schließt seine Verarbeitung ab. Dies bedeutet Folgendes:

- Beim Anzeigen von Nachrichten wird der Anzeigecursor auf die zurückgegebene Nachricht gesetzt.
- Beim Entfernen von Nachrichten wird die zurückgegebene Nachricht aus der Warteschlange entfernt.
- Sofern kein anderer Fehler auftritt, wird der Ursachencode MQRC_TRUNCATED_MSG_ACCEPTED zurückgegeben.

Ohne diese Option wird der Puffer dennoch bis zur Kapazitätsgrenze mit möglichst viel Nachrichteninhalt gefüllt. Es wird ein Warnbeendigungscode ausgegeben, die Verarbeitung wird jedoch nicht abgeschlossen. Dies bedeutet Folgendes:

- Beim Browsing von Nachrichten rückt der Anzeigecursor nicht vor.
- Beim Entfernen von Nachrichten wird die zurückgegebene Nachricht nicht aus der Warteschlange entfernt.
- Sofern kein anderer Fehler auftritt, wird der Ursachencode MQRC_TRUNCATED_MSG_FAILED zurückgegeben.

MQGMO_CONVERT
Diese Option konvertiert die Anwendungsdaten in der Nachricht, damit sie den Werten für *CodedCharSetId* und *Encoding* entsprechen, die im Parameter *MsgDesc* beim MQGET-Aufruf angegeben sind. Die Daten werden konvertiert, bevor sie in den Parameter *Buffer* kopiert werden.

Der beim Einreihen der Nachricht angegebene Wert des Feldes *Format* wird vom Konvertierungsprozess übernommen, um die Gattung der Daten in der Nachricht ermitteln zu können. Bei integrierten Formaten werden die Nachrichtendaten vom Warteschlangenmanager konvertiert, während die Konvertierung bei anderen Formaten durch einen benutzerdefinierten Exit erfolgt. Weitere Informationen zum Datenkonvertierungsexit finden Sie im Abschnitt „Datenkonvertierungsexit” auf Seite 908.

- Bei einer erfolgreichen Konvertierung wurden die Werte in den Feldern *CodedCharSetId* und *Encoding*, die im Parameter *MsgDesc* angegeben sind, bei der Rückgabe durch den MQGET-Aufruf nicht geändert.
- Wenn nur die Konvertierung fehlschlägt, werden die Nachrichtendaten unkonvertiert zurückgegeben. Die Felder *CodedCharSetId* und *Encoding* in *MsgDesc* werden auf die Werte der unkonvertierten Nachricht gesetzt. In diesem Fall wird der Beendigungscode MQCC_WARNING zurückgegeben.

In jedem Fall beschreiben diese Felder die Zeichensatzkennung und Codierung der Nachrichtendaten, die im Parameter *Buffer* zurückgegeben werden.

Die Beschreibung des Felds *Format* im Abschnitt „MQMD - Nachrichtendeskriptor” auf Seite 399 enthält eine Liste mit Formatnamen, für die der Warteschlangenmanager die Konvertierung vornimmt.

Gruppen- und Segmentoptionen: Die folgenden Optionen beziehen sich auf die Verarbeitung von Nachrichten in Gruppen und Segmenten von logischen Nachrichten. Lesen Sie vor den Optionsbeschreibungen zunächst die folgenden Definitionen wichtiger Begriffe:

Physische Nachricht
Eine physische Nachricht ist die kleinste Informationseinheit, die in eine Warteschlange gestellt oder aus einer Warteschlange entfernt werden kann. Sie entspricht häufig der Information, die in einem einzelnen MQPUT-, MQPUT1- oder MQGET-Aufruf angegeben oder abgerufen wird. Jede physische Nachricht besitzt einen eigenen Nachrichtendeskriptor (MQMD). In der Regel unterscheiden sich physische Nachrichten durch verschiedene Werte für die Nachrichten-ID voneinander, die im Feld *MsgId* im MQMD angegeben sind. Der Warteschlangenmanager erzwingt jedoch keine verschiedenen Werte.

Logische Nachricht
Eine logische Nachricht ist eine einzelne Anwendungsinformationseinheit. Bleiben Systembedingungen unberücksichtigt, ist eine logische Nachricht dasselbe wie eine physische Nachricht. Wenn logische Nachrichten sehr groß sind, kann es aufgrund von Systemeinschränkungen empfehlenswert oder erforderlich sein, eine logische Nachricht in zwei oder mehr physische Nachrichten aufzuteilen, die als Segmente bezeichnet werden.

Eine logische Nachricht, die in Segmente aufgeteilt wurde, besteht aus zwei oder mehr physischen Nachrichten mit derselben Gruppen-ID ungleich null (Feld *GroupId* im MQMD). Sie haben dieselbe Nachrichtenfolgenummer, die im Feld *MsgSeqNumber* im MQMD angegeben ist. Die Segmente unterscheiden sich durch verschiedene Werte für die relative Position des Segments (Segmentoffset) voneinander (Feld *Offset* im MQMD). Der Segmentoffset ist die relative Position der Daten in der physischen Nachricht ab Beginn der Daten in der logischen Nachricht. Da jedes Segment eine physische Nachricht ist, haben die Segmente in einer logischen Nachricht normalerweise unterschiedliche Nachrichten-IDs.

Eine logische Nachricht, die nicht in Segmente aufgeteilt wurde, aber für die die sendende Anwendung eine Segmentierung zugelassen hat, besitzt ebenfalls eine Gruppen-ID ungleich null. In diesem Fall gibt es nur eine einzige physische Nachricht mit dieser Gruppen-ID, wenn die logische Nachricht nicht zu einer Nachrichtengruppe gehört. Logische Nachrichten, für die die sendende Anwendung eine Segmentierung unterdrückt hat, besitzen eine Gruppen-ID mit dem Wert null (MQGI_NONE), außer wenn die logische Nachricht zu einer Nachrichtengruppe gehört.

Nachrichtengruppe
Eine Nachrichtengruppe ist eine Gruppe mit einer logischen Nachricht oder mehreren logischen Nachrichten, die dieselbe Gruppen-ID ungleich null besitzen. Die logischen Nachrichten in der Gruppe unterscheiden sich durch verschiedene Werte für die Nachrichtenfolgenummer. Die Folgenummer ist eine Ganzzahl zwischen 1 und n. Dabei steht "n" für die Anzahl der logischen Nachrichten in der Gruppe. Wenn eine oder mehrere logische Nachrichten segmentiert sind, enthält die Gruppe mehr als n physische Nachrichten.

MQGMO_LOGICAL_ORDER
MQGMO_LOGICAL_ORDER steuert die Reihenfolge, in der Nachrichten von aufeinanderfolgenden MQGET-Aufrufen für die Warteschlangenkennung zurückgegeben werden. Die Option muss bei jedem Aufruf angegeben werden.

Wird für dieselbe Warteschlangenkennung die Option MQGMO_LOGICAL_ORDER für aufeinanderfolgende MQGET-Aufrufe angegeben, werden Nachrichten, die sich in Gruppen befinden, in der Reihenfolge ihrer Nachrichtenfolgenummern zurückgegeben. Segmente logischer Nachrichten werden in der Reihenfolge zurückgegeben, die sich aus ihren jeweiligen Segmentoffsets ergibt. Diese Reihenfolge kann von der Reihenfolge abweichen, in der sich diese Nachrichten und Segmente in der Warteschlange befinden.

Anmerkung: Die Angabe von MQGMO_LOGICAL_ORDER hat keine negativen Folgen für Nachrichten, die keinen Gruppen angehören. Dies gilt auch für Nachrichten, die keine Segmente sind. Tatsächlich werden diese Nachrichten so behandelt, als gehörten sie zu einer Nachrichtengruppe, die nur eine Nachricht enthalten. Die Option MQGMO_LOGICAL_ORDER kann beim Abruf von Nachrichten aus Warteschlangen, die eine Kombination aus gruppierten Nachrichten, Nachrichtensegmenten und nicht gruppierten und nicht segmentierten Nachrichten enthalten, gefahrlos angegeben werden.

Damit die Nachrichten in der erforderlichen Reihenfolge zurückgegeben werden, merkt sich der Warteschlangenmanager die Gruppen- und Segmentinformationen über mehrere aufeinanderfolgende MQGET-Aufrufe hinweg. Die Gruppen- und Segmentinformationen geben die aktuelle Nachrichtengruppe und die aktuelle logische Nachricht für die Warteschlangenkennung an. Sie geben außerdem die aktuelle Position innerhalb der Gruppe und der logischen Nachricht an und informieren Sie darüber, ob die Nachrichten innerhalb einer Arbeitseinheit abgerufen werden. Da sich der Warteschlangenmanager diese Informationen merkt, muss die Anwendung die Gruppen- und Segmentinformationen nicht vor jedem MQGET-Aufruf festlegen. Dies bedeutet also, dass die Anwendung die Felder *GroupId*, *MsgSeqNumber* und *Offset* im MQMD nicht festlegen muss. Allerdings muss die Anwendung die Option MQGMO_SYNCPOINT bzw. MQGMO_NO_SYNCPOINT bei jedem Aufruf ordnungsgemäß festlegen.

Wenn die Warteschlange geöffnet wird, ist keine aktuelle Nachrichtengruppe und keine aktuelle logische Nachricht verfügbar. Eine Nachrichtengruppe wird zur aktuellen Nachrichtengruppe, wenn eine Nachricht mit dem Flag MQMF_MSG_IN_GROUP vom MQGET-Aufruf zurückgegeben wird. Ist die Option MQGMO_LOGICAL_ORDER für aufeinanderfolgende Aufrufe angegeben, bleibt die betreffende Gruppe die aktuelle Gruppe, bis eine Nachricht mit folgenden Einstellungen zurückgegeben wird:

- MQMF_LAST_MSG_IN_GROUP ohne MQMF_SEGMENT (das heißt, die letzte logische Nachricht in der Gruppe ist nicht in Segmente aufgeteilt) oder
- MQMF_LAST_MSG_IN_GROUP mit MQMF_LAST_SEGMENT (das heißt, die zurückgegebene Nachricht ist das letzte Segment der letzten logischen Nachricht in der Gruppe).

Wenn eine solche Nachricht zurückgegeben wird, wird die Nachrichtengruppe beendet, und bei einer erfolgreichen Beendigung des MQGET-Aufrufs ist keine aktuelle Gruppe mehr vorhanden. Auf ähnliche Weise wird eine logische Nachricht zur aktuellen logischen Nachricht, wenn eine Nachricht mit dem Flag MQMF_SEGMENT vom MQGET -Aufruf zurückgegeben wird. Die logische Nachricht wird beendet, wenn die Nachricht mit dem Flag MQMF_LAST_SEGMENT zurückgegeben wird.

Sind keine Auswahlkriterien angegeben, geben aufeinanderfolgende MQGET-Aufrufe die Nachrichten für die erste Nachrichtengruppe in der Warteschlange in der korrekten Reihenfolge zurück. Anschließend geben sie die Nachrichten für die zweite Nachrichtengruppe usw. zurück, bis keine Nachrichten mehr verfügbar sind. Durch die gezielte Angabe von mindestens einer der folgenden Optionen im Feld *MatchOptions* können bestimmte Nachrichtengruppen für die Rückgabe ausgewählt werden:

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Diese Optionen sind jedoch nur wirksam, wenn keine aktuelle Nachrichtengruppe oder logische Nachricht vorhanden ist. Weitere Informationen finden Sie in der Beschreibung des Felds *MatchOptions* in „MQGMO – Nachrichtenabrufoptionen” auf Seite 348 .

In der Tabelle 506 auf Seite 369 werden die Werte der Felder *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* und *Offset* angegeben, nach denen der Warteschlangenmanager bei der Ermittlung einer Nachricht für die Rückgabe durch den MQGET-Aufruf sucht. Die Regeln gelten sowohl für das Entfernen von Nachrichten aus der Warteschlange als auch für das Browsing von Nachrichten in der Warteschlange. In der Tabelle bedeutet "Beliebig", dass "Ja" oder "Nein" möglich ist:

LOG ORD
Gibt an, ob die Option MQGMO_LOGICAL_ORDER im Aufruf angegeben ist.

Cur grp
Gibt an, ob vor dem Aufruf eine aktuelle Nachrichtengruppe vorhanden ist.

Cur log msg
Gibt an, ob vor dem Aufruf eine aktuelle logische Nachricht existiert.

Sonstige Spalten
Geben die Werte an, nach denen der Warteschlangenmanager sucht. Die Angabe "Vorherig" bezeichnet den Feldwert, der in der vorherigen Nachricht für die Warteschlangenkennung zurückgegeben wurde.

Tabelle 506. MQGET-Optionen für Nachrichten in Gruppen und Segmenten von logischen Nachrichten

Angegebene Option	Gruppen- und log-msg-Status vor dem Aufruf		Werte, nach denen der Warteschlangenmanager sucht				
LOG ORD	**Cur grp**	**Cur log msg**	*MsgId*	*CorrelId*	*GroupId*	*MsgSeqNumber*	*Offset*

Tabelle 506. MQGET-Optionen für Nachrichten in Gruppen und Segmenten von logischen Nachrichten (Forts.)

Angegebene Option	Gruppen- und log-msg-Status vor dem Aufruf		Werte, nach denen der Warteschlangenmanager sucht				
Ja	Nein	Nein	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert	1	0
Ja	Nein	Ja	Jede Nachrichten-ID	Jede Korrelations-ID	Vorherige Gruppen-ID	1	Voriger Offset + vorige Segmentlänge
Ja	Ja	Nein	Jede Nachrichten-ID	Jede Korrelations-ID	Vorherige Gruppen-ID	Vorige Folgenummer + 1	0
Ja	Ja	Ja	Jede Nachrichten-ID	Jede Korrelations-ID	Vorherige Gruppen-ID	Vorige Folgenummer	Voriger Offset + vorige Segmentlänge
Nein	Eines	Eines	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert	Durch *MatchOptions* gesteuert

Wenn die Warteschlange mehrere Nachrichtengruppen enthält, die für die Rückgabe infrage kommen, werden die Gruppen in der Reihenfolge zurückgegeben, die durch die Position des ersten Segments der ersten logischen Nachricht in jeder Gruppe in der Warteschlange bestimmt wird. Die physischen Nachrichten, die Nachrichtenfolgenummern mit dem Wert "1" und relative Positionen mit dem Wert "0" haben, bestimmen also die Reihenfolge, in der infrage kommende Gruppen zurückgegeben werden.

Die Option MQGMO_LOGICAL_ORDER wirkt sich wie folgt auf Arbeitseinheiten aus:

- Wenn die erste logische Nachricht oder das erste Segment einer Gruppe in einer Arbeitseinheit abgerufen wird, müssen auch alle anderen logischen Nachrichten und Segmente in einer Arbeitseinheit abgerufen werden, wenn dieselbe Warteschlangenkennung verwendet wird. Sie müssen jedoch nicht in derselben Arbeitseinheit abgerufen werden. Dadurch kann eine Nachrichtengruppe mit vielen physischen Nachrichten auf mehrere aufeinanderfolgende Arbeitseinheiten für die Warteschlangenkennung aufgeteilt werden.
- Wird die erste logische Nachricht oder das erste Segment in einer Gruppe *nicht* innerhalb einer Arbeitseinheit abgerufen und wird dieselbe Warteschlangenkennung verwendet, können keine der übrigen logischen Nachrichten und Segmente in der Gruppe innerhalb einer Arbeitseinheit abgerufen werden.

Werden diese Bedingungen nicht erfüllt, schlägt der MQGET-Aufruf mit dem Ursachencode MQRC_INCONSISTENT_UOW fehl.

Wenn MQGMO_LOGICAL_ORDER angegeben wird, darf der Wert für MQGMO , der im MQGET -Aufruf bereitgestellt wird, nicht kleiner als MQGMO_VERSION_2und der Wert für MQMD nicht kleiner als MQMD_VERSION_2sein. Wenn diese Bedingung nicht erfüllt ist, schlägt der Aufruf mit Ursachencode MQRC_WRONG_GMO_VERSION bzw. MQRC_WRONG_MD_VERSION fehl.

Wenn MQGMO_LOGICAL_ORDER für aufeinanderfolgende MQGET -Aufrufe für die Warteschlangenkennung *nicht* angegeben wird, werden Nachrichten unabhängig davon zurückgegeben, ob sie zu Nachrichtengruppen gehören oder ob sie Segmente logischer Nachrichten sind. Dies bedeutet, dass Nachrichten oder Segmente aus einer bestimmten Gruppe oder logischen Nachricht möglicherweise nicht in der richtigen Reihenfolge zurückgegeben werden oder mit Nachrichten oder Segmenten aus anderen Gruppen oder logischen Nachrichten bzw. mit Nachrichten, die keinen Gruppen angehören und keine Segmente sind, vermischt werden. In dieser Situation kann die gezielte Rückgabe bestimmter Nachrichten durch aufeinanderfolgende MQGET-Aufrufe mit den Optionen des Typs MQMO_* gesteuert werden, die für die betreffenden Aufrufe angegeben werden (diese Optionen werden in der Beschrei-

bung des Felds *MatchOptions* im Abschnitt „MQGMO – Nachrichtenabrufoptionen” auf Seite 348 ausführlich erläutert).

Diese Technik kann für den Neustart einer Nachrichtengruppe oder logischen Nachricht während der Verarbeitung verwendet werden, nachdem ein Systemfehler aufgetreten ist. Beim Systemwiederanlauf kann die Anwendung die Felder *GroupId*, *MsgSeqNumber*, *Offset* und *MatchOptions* auf die geeigneten Werte setzen und anschließend den MQGET-Aufruf mit der Option MQGMO_SYNCPOINT oder MQGMO_NO_SYNCPOINT, jedoch *ohne* Angabe der Option MQGMO_LOGICAL_ORDER, erneut ausgeben. Verläuft dieser Aufruf erfolgreich, behält der Warteschlangenmanager die Gruppen- und Segmentinformationen bei. Nachfolgende MQGET-Aufrufe, die diese Warteschlangenkennung verwenden, können die Option MQGMO_LOGICAL_ORDER wie gewohnt angeben.

Die vom Warteschlangenmanager für den MQGET-Aufruf beibehaltenen Gruppen- und Segmentinformationen werden gesondert von den Gruppen- und Segmentinformationen gespeichert, die für den MQPUT-Aufruf beibehalten werden. Darüber hinaus behält der Warteschlangenmanager die folgenden Informationen bei:

- MQGET-Aufrufe, die Nachrichten aus der Warteschlange entfernen.
- MQGET-Aufrufe, die Nachrichten in der Warteschlange durchsuchen.

Für jede angegebene Warteschlangenkennung kann die Anwendung MQGET-Aufrufe, bei denen MQGMO_LOGICAL_ORDER angegeben ist, mit MQGET-Aufrufen kombinieren, bei denen dies nicht der Fall ist. Beachten Sie jedoch die folgenden Aspekte:

- Wenn Sie die Option MQGMO_LOGICAL_ORDER übergehen, bewirkt jeder erfolgreiche MQGET-Aufruf, dass der Warteschlangenmanager die gespeicherten Gruppen- und Segmentinformationen auf die Werte setzt, die der zurückgegebenen Nachricht entsprechen; dabei werden die vorhandenen Gruppen- und Segmentinformationen ersetzt, die vom Warteschlangenmanager für die Warteschlangenkennung gespeichert wurden. Es werden nur die Informationen geändert, die der Aktion des Aufrufs (Durchsuchen oder Entfernen) entsprechen.
- Wenn Sie die Option MQGMO_LOGICAL_ORDER übergehen, schlägt der Aufruf nicht fehl, falls eine aktuelle Nachrichtengruppe oder logische Nachricht vorhanden ist; der Aufruf kann mit dem Code MQCC_WARNING erfolgreich ausgeführt werden. Tabelle 507 auf Seite 371 gibt die verschiedenen Fälle an, die auftreten können. Wenn der Beendigungscode in diesen Fällen nicht MQCC_OK lautet, wird als Ursachencode einer der folgenden Werte gemeldet (je nach Situation):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_UOW

  Anmerkung: Beim Durchsuchen einer Warteschlange oder beim Schließen einer Warteschlange, die nicht für eine Eingabe, sondern für das Browsing geöffnet wurde, überprüft der Warteschlangenmanager die Gruppen- und Segmentinformationen nicht; in diesen Fällen lautet der Beendigungscode immer MQCC_OK (sofern keine sonstigen Fehler vorliegen).

Tabelle 507. Ergebnis, wenn der MQGET- oder MQCLOSE-Aufruf nicht mit den Gruppen- und Segmentinformationen konsistent ist

Aktueller Aufruf ist	Der vorherige Aufruf lautete MQGET mit der Option MQGMO_LOGICAL_ORDER	Der vorherige Aufruf lautete MQGET ohne die Option MQGMO_LOGICAL_ORDER
MQGET mit der Option MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET ohne die Option MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE mit nicht beendeter Gruppe oder logischen Nachricht	MQCC_WARNING	MQCC_OK

Für Anwendungen, die Nachrichten und Segmente in logischer Reihenfolge abrufen möchten, wird empfohlen, die Option MQGMO_LOGICAL_ORDER anzugeben, da die Verwendung dieser Option am einfachsten ist. Bei dieser Option muss die Anwendung die Gruppen- und Segmentinformationen nicht verwalten, da der Warteschlangenmanager diese Informationen verwaltet. Es kann jedoch vorkommen, dass bestimmte Fachanwendungen eine höhere Kontrolle benötigen, als mit der Option MQGMO_LOGICAL_ORDER möglich ist. Dies kann durch die Nichtangabe dieser Option erreicht werden. Die Anwendung muss dann sicherstellen, dass die Felder *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*und *Offset* in MQMDund die Optionen MQMO_* in *MatchOptions* in MQGMOvor jedem MQGET -Aufruf ordnungsgemäß festgelegt sind.

Eine Anwendung, die beispielsweise physische Nachrichten, die sie empfängt, *weiterleiten* möchte, ohne zu berücksichtigen, ob sich diese Nachrichten in Gruppen oder Segmenten logischer Nachrichten befinden, darf *nicht* MQGMO_LOGICAL_ORDERangeben. In einem komplexen Netz aus verschiedenen Pfaden zwischen sendenden und empfangenden Warteschlangenmanagern kann es dazu kommen, dass physische Nachrichten nicht in der richtigen Reihenfolge eintreffen. Wenn die weiterleitende Anwendung weder MQGMO_LOGICAL_ORDER noch die entsprechende Option MQPMO_LOGICAL_ORDER im MQPUT-Aufruf angibt, kann sie jede physische Nachricht abrufen und weiterleiten, sobald diese eintrifft. Sie muss nicht warten, bis die nächste Nachricht in der logischen Reihenfolge eintrifft.

Sie können MQGMO_LOGICAL_ORDER zusammen mit jeder anderen Option des Typs MQGMO_* angeben, sowie je nach Situation (siehe oben) mit verschiedenen Optionen des Typs MQMO_*.

- Unter z/OS wird diese Option für private und gemeinsam genutzte Warteschlangen unterstützt, allerdings muss die Warteschlange den Indextyp MQIT_GROUP_ID aufweisen. Bei gemeinsam genutzten Warteschlangen muss das CFSTRUCT-Objekt, dem die Warteschlange zugeordnet wird, den Stand CFLEVEL(3) oder CFLEVEL(4) haben.
- Unter AIX, HP-UX, IBM i, Solaris, Linux, Windows und WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind, wird diese Option für alle lokalen Warteschlangen unterstützt.

MQGMO_COMPLETE_MSG

Nur vollständige logische Nachrichten können vom MQGET-Aufruf zurückgegeben werden. Ist die logische Nachricht in Segmente aufgeteilt, fügt der Warteschlangenmanager die Segmente wieder zusammen und gibt die vollständige logische Nachricht an die Anwendung zurück; für die abrufende Anwendung ist nicht mehr erkennbar, dass die logische Nachricht in Segmente aufgeteilt war.

Anmerkung: Nur diese Option bewirkt, dass der Warteschlangenmanager Nachrichtensegmente neu erstellt. Ist sie nicht angegeben, werden Segmente einzeln an die Anwendung zurückgegeben, falls sie in der Warteschlange vorhanden sind (und die übrigen Auswahlkriterien erfüllen, die im MQGET-Aufruf festgelegt sind). Anwendungen, die keine einzelnen Segmente empfangen möchten, müssen die Option MQGMO_COMPLETE_MSG immer angeben.

Damit diese Option verwendet werden kann, muss die Anwendung einen Puffer bereitstellen, der für die vollständige Nachricht ausreicht, oder die Option MQGMO_ACCEPT_TRUNCATED_MSG angeben.

Wenn die Warteschlange segmentierte Nachrichten enthält, bei denen einige Segmente fehlen (weil sie beispielsweise durch das Netz verzögert wurden und noch nicht eingetroffen sind), wird durch die Angabe der Option MQGMO_COMPLETE_MSG der Abruf von Segmenten verhindert, die zu unvollständigen logischen Nachrichten gehören. Diese Nachrichtensegmente tragen jedoch zum Wert des Warteschlangenattributs *CurrentQDepth* bei. Dies bedeutet, dass möglicherweise keine abrufbare logische Nachricht vorhanden ist, auch wenn *CurrentQDepth* größer als null ist.

Bei *persistenten* Nachrichten kann der Warteschlangenmanager die Segmente nur innerhalb einer Arbeitseinheit wieder zusammenfügen:

- Wird der MQGET-Aufruf innerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt, wird diese Arbeitseinheit verwendet. Schlägt der Aufruf aufgrund des Neuerstellungsprozesses fehl, stellt der Warteschlangenmanager in der Warteschlange alle Segmente wieder her, die während der Neuerstellung entfernt wurden. Allerdings verhindert das Fehlschlagen nicht, dass die Arbeitseinheit erfolgreich festgeschrieben wird.

- Wird der Aufruf außerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt und ist keine benutzerdefinierte Arbeitseinheit vorhanden, wird sie vom Warteschlangenmanager für die Dauer des Aufrufs erstellt. Ist der Aufruf erfolgreich, schreibt der Warteschlangenmanager die Arbeitseinheit automatisch fest (dies muss also nicht durch die Anwendung erfolgen). Wenn der Aufruf fehlschlägt, setzt der Warteschlangenmanager die Arbeitseinheit zurück.
- Wird der Aufruf außerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt und ist eine benutzerdefinierte Arbeitseinheit vorhanden, kann der Warteschlangenmanager keine erneute Zusammenfügung vornehmen. Wenn für die Nachricht keine Neuerstellung erforderlich ist, kann der Aufruf dennoch erfolgreich durchgeführt werden. Andernfalls schlägt der Aufruf mit dem Ursachencode MQRC_UOW_NOT_AVAILABLE fehl.

Bei *nicht persistenten* Nachrichten benötigt der Warteschlangenmanager für die erneute Zusammenfügung keine verfügbare Arbeitseinheit.

Jede physische Nachricht, bei der es sich um ein Segment handelt, hat einen eigenen Nachrichtendeskriptor. Bei Segmenten, die eine einzelne logische Nachricht ergeben, sind die meisten Felder im Nachrichtendeskriptor für alle Segmente in der logischen Nachricht identisch; in der Regel weisen nur die Felder *MsgId*, *Offset* und *MsgFlags* unterschiedliche Werte für die Segmente in der logischen Nachricht auf. Wird ein Segment jedoch von einem zwischengeschalteten Warteschlangenmanager in eine Warteschlange für nicht zustellbare Nachrichten gestellt, ruft der Handler der Warteschlange für nicht zustellbare Nachrichten die Nachricht unter Angabe der Option MQGMO_CONVERT ab. Dies kann dazu führen, dass der Zeichensatz oder die Codierung des Segments geändert wird. Wenn der Handler der Warteschlange für nicht zustellbare Nachrichten das Segment auf diese Weise sendet, kann das Segment einen Zeichensatz oder eine Codierung aufweisen, die sich von den anderen Segmenten in der logischen Nachricht unterscheidet, wenn es beim Zielwarteschlangenmanager eintrifft.

Eine logische Nachricht, die sich aus Segmenten zusammensetzt, deren Werte in den Feldern *CodedCharSetId* und *Encoding* unterschiedlich sind, kann nicht vom Warteschlangenmanager wieder zu einer einzelnen logischen Nachricht zusammengefügt werden. Stattdessen fügt der Warteschlangenmanager die Nachricht zusammen und gibt die ersten aufeinanderfolgenden Segmente am Anfang der logischen Nachricht zurück, die dieselben Zeichensätze und Codierungen aufweisen. In diesem Fall wird der MQGET-Aufruf mit dem Beendigungscode MQCC_WARNING und (je nach Situation) mit dem Ursachencode MQRC_INCONSISTENT_CCSIDS oder MQRC_INCONSISTENT_ENCODINGS beendet. Dabei spielt es keine Rolle, ob die Option MQGMO_CONVERT angegeben wurde oder nicht. Zum Abruf der verbleibenden Segmente muss die Anwendung den MQGET-Aufruf ohne die Option MQGMO_COMPLETE_MSG ausgeben, damit die Segmente einzeln abgerufen werden können. Mit der Option MQGMO_LOGICAL_ORDER können die verbleibenden Segmente in der richtigen Reihenfolge nacheinander abgerufen werden.

Eine Anwendung, die Segmente einstellt, kann auch sonstige Felder im Nachrichtendeskriptor auf Werte setzen, die sich bei den verschiedenen Segmenten unterscheiden. Dies stellt jedoch keinen Vorteil dar, wenn die empfangende Anwendung die logische Nachricht mit der Option MQGMO_COMPLETE_MSG abruft. Wenn der Warteschlangenmanager eine logische Nachricht wieder zusammenfügt, gibt er im Nachrichtendeskriptor die Nachrichtendeskriptorwerte des *ersten* Segments zurück; ausgenommen hiervon ist lediglich das Feld *MsgFlags*, durch dessen Festlegung der Warteschlangenmanager angibt, dass es sich bei der zusammengefügten Nachricht um das einzige Segment handelt.

Wird MQGMO_COMPLETE_MSG für eine Berichtsnachricht angegeben, führt der Warteschlangenmanager eine besondere Verarbeitung aus. Der Warteschlangenmanager überprüft die Warteschlange darauf, ob alle Berichtsnachrichten des Berichtstyps, der sich auf die unterschiedlichen Segmente in der logischen Nachricht bezieht, vorhanden sind. Ist dies der Fall, können sie durch die Angabe von MQGMO_COMPLETE_MSG als einzelne Nachricht abgerufen werden. Damit dies möglich ist, müssen die Berichtsnachrichten entweder von einem Warteschlangenmanager oder Nachrichtenkanalagenten, der die Segmentierung unterstützt, generiert werden, oder die einleitende Anwendung muss mindestens 100 Bytes an Nachrichtendaten anfordern (das heißt, die Optionen MQRO_*_WITH_DATA oder MQRO_*_WITH_FULL_DATA müssen entsprechend angegeben werden). Wenn nicht alle Anwendungsdaten für ein Segment vorhanden sind, werden die fehlenden Byte in der zurückgegebenen Berichtsnachricht durch Nullen ersetzt.

Wenn MQGMO_COMPLETE_MSG mit MQGMO_MSG_UNDER_CURSOR oder MQGMO_BROWSE_MSG_UNDER_CURSOR angegeben wird, muss der Anzeigecursor auf einer Nachricht positioniert werden, deren Feld *Offset* in MQMD den Wert 0 hat. Wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_INVALID_MSG_UNDER_CURSOR fehl.

Da MQGMO_COMPLETE_MSG die Option MQGMO_ALL_SEGMENTS_AVAILABLE einschließt, muss diese nicht angegeben werden.

MQGMO_COMPLETE_MSG kann mit Ausnahme von MQGMO_SYNCPOINT_IF_PERSISTENT in Verbindung mit allen anderen Optionen des Typs MQGMO_*, sowie mit Ausnahme von MQMO_MATCH_OFFSET in Verbindung mit allen Optionen des Typs MQMO_* angegeben werden.

- Unter z/OS wird diese Option für private und gemeinsam genutzte Warteschlangen unterstützt, allerdings muss die Warteschlange den Indextyp MQIT_GROUP_ID aufweisen. Bei gemeinsam genutzten Warteschlangen muss das CFSTRUCT-Objekt, dem die Warteschlange zugeordnet wird, den Stand CFLEVEL(3) oder CFLEVEL(4) haben.
- Unter AIX, HP-UX, IBM i, Solaris, Linux, Windows und WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind, wird diese Option für alle lokalen Warteschlangen unterstützt.

MQGMO_ALL_MSGS_AVAILABLE
Die Nachrichten in einer Gruppe stehen nur für den Abruf zur Verfügung, wenn *alle* Nachrichten in der Gruppe verfügbar sind. Wenn die Warteschlange Nachrichtengruppen enthält, bei denen einige Nachrichten fehlen (weil sie beispielsweise durch das Netz verzögert wurden und noch nicht eingetroffen sind), wird durch die Angabe der Option MQGMO_ALL_MSGS_AVAILABLE der Abruf von Nachrichten verhindert, die zu unvollständigen Gruppen gehören. Diese Nachrichten werden allerdings dennoch in den Wert des Warteschlangenattributs *CurrentQDepth* einbezogen, was bedeutet, dass möglicherweise keine abrufbaren Nachrichtengruppen vorhanden sind, obwohl das Feld *CurrentQDepth* einen größeren Wert als null aufweist. Wenn keine anderen Nachrichten abrufbar sind, wird der Ursachencode MQRC_NO_MSG_AVAILABLE zurückgegeben, sobald das angegebene Warteintervall (falls vorhanden) abgelaufen ist.

Die Verarbeitung von MQGMO_ALL_MSGS_AVAILABLE hängt davon ab, ob MQGMO_LOGICAL_ORDER ebenfalls angegeben ist:

- Wenn beide Optionen angegeben werden, wirkt sich die Option MQGMO_ALL_MSGS_AVAILABLE *nur* aus, wenn keine aktuelle Gruppe oder logische Nachricht vorhanden ist. Falls eine aktuelle Gruppe oder logische Nachricht vorhanden *ist*, wird MQGMO_ALL_MSGS_AVAILABLE ignoriert. Dies bedeutet, dass MQGMO_ALL_MSGS_AVAILABLE aktiviert bleiben kann, wenn Nachrichten in logischer Reihenfolge verarbeitet werden.
- Wird MQGMO_ALL_MSGS_AVAILABLE ohne die Option MQGMO_LOGICAL_ORDER angegeben, wirkt sich MQGMO_ALL_MSGS_AVAILABLE *immer* aus. Dies bedeutet, dass die Option deaktiviert werden muss, nachdem die erste Nachricht der Gruppe aus der Warteschlange entfernt wurde, um die übrigen Nachrichten der Gruppe entfernen zu können.

Wird ein MQGET-Aufruf mit der Option MQGMO_ALL_MSGS_AVAILABLE erfolgreich beendet, bedeutet dies, dass sich zum Zeitpunkt der Ausgabe des MQGET-Aufrufs alle Nachrichten in der Gruppe in der Warteschlange befanden. Allerdings müssen Sie beachten, dass andere Anwendungen nach wie vor Nachrichten aus der Gruppe entfernen können (die Gruppe ist nicht für die Anwendung gesperrt, die die erste Nachricht in der Gruppe abruft).

Wenn Sie diese Option übergehen, können Nachrichten, die Gruppen angehören, auch dann abgerufen werden, wenn die Gruppe unvollständig ist.

Da MQGMO_ALL_MSGS_AVAILABLE die Option MQGMO_ALL_SEGMENTS_AVAILABLE einschließt, muss diese nicht angegeben werden.

MQGMO_ALL_MSGS_AVAILABLE kann mit jeder anderen Option des Typs MQGMO_* sowie mit allen Optionen des Typs MQMO_* angegeben werden.

- Unter z/OS wird diese Option für private und gemeinsam genutzte Warteschlangen unterstützt, allerdings muss die Warteschlange den Indextyp MQIT_GROUP_ID aufweisen. Bei gemeinsam ge-

nutzten Warteschlangen muss das CFSTRUCT-Objekt, dem die Warteschlange zugeordnet wird, den Stand CFLEVEL(3) oder CFLEVEL(4) haben.

- Unter AIX, HP-UX, IBM i, Solaris, Linux, Windows und WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind, wird diese Option für alle lokalen Warteschlangen unterstützt.

MQGMO_ALL_SEGMENTS_AVAILABLE
Die Segmente in einer logischen Nachricht stehen nur für den Abruf zur Verfügung, wenn *alle* Segmente in der logischen Nachricht verfügbar sind. Wenn die Warteschlange segmentierte Nachrichten enthält, bei denen einige Segmente fehlen (weil sie beispielsweise durch das Netz verzögert wurden und noch nicht eingetroffen sind), wird durch die Angabe der Option MQGMO_ALL_SEGMENTS_AVAILABLE der Abruf von Segmenten verhindert, die zu unvollständigen logischen Nachrichten gehören. Diese Segmente werden allerdings dennoch in den Wert des Warteschlangenattributs *CurrentQDepth* einbezogen, was bedeutet, dass möglicherweise keine abrufbaren logischen Nachrichten vorhanden sind, obwohl das Feld *CurrentQDepth* einen größeren Wert als null aufweist. Wenn keine anderen Nachrichten abrufbar sind, wird der Ursachencode MQRC_NO_MSG_AVAILABLE zurückgegeben, sobald das angegebene Warteintervall (falls vorhanden) abgelaufen ist.

Die Verarbeitung von MQGMO_ALL_SEGMENTS_AVAILABLE hängt davon ab, ob MQGMO_LOGICAL_ORDER ebenfalls angegeben ist:

- Wenn beide Optionen angegeben werden, wirkt sich die Option MQGMO_ALL_SEGMENTS_AVAILABLE *nur* aus, wenn keine aktuelle logische Nachricht vorhanden ist. Falls eine aktuelle logische Nachricht vorhanden *ist*, wird MQGMO_ALL_SEGMENTS_AVAILABLE ignoriert. Dies bedeutet, dass MQGMO_ALL_SEGMENTS_AVAILABLE aktiviert bleiben kann, wenn Nachrichten in logischer Reihenfolge verarbeitet werden.
- Wird MQGMO_ALL_SEGMENTS_AVAILABLE ohne die Option MQGMO_LOGICAL_ORDER angegeben, wirkt sich MQGMO_ALL_SEGMENTS_AVAILABLE *immer* aus. Dies bedeutet, dass die Option deaktiviert werden muss, nachdem das erste Segment der logischen Nachricht aus der Warteschlange entfernt wurde, um die übrigen Segmente der logischen Nachricht entfernen zu können.

Wenn diese Option nicht angegeben wird, können Nachrichtensegmente auch dann abgerufen werden, wenn die logische Nachricht unvollständig ist.

Sowohl bei der Option MQGMO_COMPLETE_MSG als auch bei MQGMO_ALL_SEGMENTS_AVAILABLE müssen alle Segmente verfügbar sein, damit eines der Segmente abgerufen werden kann. Bei der ersten Option wird jedoch die vollständige Nachricht zurückgegeben, während bei der zweiten Option die Segmente einzeln abgerufen werden können.

Wird MQGMO_ALL_SEGMENTS_AVAILABLE für eine Berichtsnachricht angegeben, prüft der Warteschlangenmanager die Warteschlange, um festzustellen, ob für jedes der Segmente, aus denen sich die vollständige logische Nachricht zusammensetzt, mindestens eine Berichtsnachricht vorhanden ist. Ist dies der Fall, gilt die Bedingung MQGMO_ALL_SEGMENTS_AVAILABLE als erfüllt. Da der Warteschlangenmanager jedoch den *Typ* der vorhandenen Berichtsnachrichten nicht überprüft, können die Berichtsnachrichten in Zusammenhang mit den Segmenten der logischen Nachricht unterschiedliche Berichtstypen aufweisen. Folglich bedeutet der Erfolg der Option MQGMO_ALL_SEGMENTS_AVAILABLE nicht unbedingt, dass auch MQGMO_COMPLETE_MSG erfolgreich verlaufen wird. Wenn bei den Segmenten einer bestimmten logischen Nachricht eine Mischung aus verschiedenen Berichtstypen vorhanden *ist*, müssen diese Berichtsnachrichten einzeln abgerufen werden.

MQGMO_ALL_SEGMENTS_AVAILABLE kann mit jeder anderen Option des Typs MQGMO_* sowie mit allen Optionen des Typs MQMO_* angegeben werden.

- Unter z/OS wird diese Option für private und gemeinsam genutzte Warteschlangen unterstützt, allerdings muss die Warteschlange den Indextyp MQIT_GROUP_ID aufweisen. Bei gemeinsam genutzten Warteschlangen muss das CFSTRUCT-Objekt, dem die Warteschlange zugeordnet wird, den Stand CFLEVEL(3) oder CFLEVEL(4) haben.
- Unter AIX, HP-UX, IBM i, Solaris, Linux, Windows und WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind, wird diese Option für alle lokalen Warteschlangen unterstützt.

Eigenschaftsoptionen: Die folgenden Optionen beziehen sich auf die Eigenschaften der Nachricht:

MQGMO_PROPERTIES_AS_Q_DEF

Die Eigenschaften der Nachricht sollten mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung) so wie vom Warteschlangenattribut *PropertyControl* definiert dargestellt werden. Wenn im Feld *MsgHandle* eine Nachrichtenkennung angegeben ist, wird diese Option ignoriert und die Eigenschaften der Nachricht sind mit *MsgHandle* verfügbar, wenn der Wert des Warteschlangenattributs *PropertyControl* nicht MQPROP_FORCE_MQRFH2 ist.

Dies ist die Standardaktion, wenn keine Eigenschaftenoptionen angegeben sind.

MQGMO_PROPERTIES_IN_HANDLE

Die Eigenschaften der Nachricht sollten unter Verwendung der im Feld *MsgHandle* angegebenen Nachrichtenkennung zur Verfügung gestellt werden. Wenn keine Nachrichtenkennung angegeben ist, schlägt der Aufruf mit der Ursache MQRC_HMSG_ERROR fehl.

Anmerkung: Wenn die Nachricht zu einem späteren Zeitpunkt von einer Anwendung gelesen wird, die keine Nachrichtenkennung erstellt, stellt der Warteschlangenmanager beliebige Nachrichteneigenschaften in eine MQRFH2-Struktur. Möglicherweise werden Sie feststellen, dass das Vorhandensein eines nicht erwarteten MQRFH2-Headers das Verhalten einer vorhandenen Anwendung beeinträchtigt.

MQGMO_NO_PROPERTIES

Mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung) werden keine Eigenschaften der Nachricht abgerufen. Ist im Feld *MsgHandle* eine Nachrichtenkennung angegeben, wird sie ignoriert.

MQGMO_PROPERTIES_FORCE_MQRFH2

Die Eigenschaften der Nachricht sollten mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung) mithilfe der MQRFH2-Header dargestellt werden. Dadurch wird für Anwendungen, die das Abrufen von Eigenschaften erwarten, jedoch nicht so geändert werden können, dass sie Nachrichtenkennungen verwenden, die Kompatibilität mit älteren Versionen gewährleistet. Wenn ein Nachrichtenhandle (*MsgHandle*) angegeben ist, wird es ignoriert.

MQGMO_PROPERTIES_COMPATIBILITY
Wenn die Nachricht eine Eigenschaft mit dem Präfix **"mcd."**, **"jms."**, **"usr."**oder **"mqext."**enthält, werden alle Nachrichteneigenschaften der Anwendung in einem MQRFH2 -Header zugestellt. Andernfalls werden alle Eigenschaften der Nachricht, außer denen, die im Nachrichtendeskriptor (oder Erweiterung) enthalten sind, gelöscht und sind nicht mehr für die Anwendung verfügbar.

Standardoption: Wenn keine der zuvor beschriebenen Optionen erforderlich ist, kann die folgende Option verwendet werden:

MQGMO_NONE
Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an. MQGMO_NONE unterstützt die Programmdokumentation und ist nicht für die Verwendung mit anderen Optionen vorgesehen. Da sie jedoch den Wert null hat, kann eine derartige Verwendung nicht erkannt werden.

Der Anfangswert des Felds *Options* lautet MQGMO_NO_WAIT plus MQGMO_PROPERTIES_AS_Q_DEF.

Reserved1 (MQCHAR)

Dies ist ein reserviertes Feld. Der Anfangswert dieses Felds ist ein Leerzeichen. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.

Reserved2 (MQLONG)

Dies ist ein reserviertes Feld. Der Anfangswert dieses Felds ist ein Leerzeichen. Dieses Feld wird ignoriert, wenn *Version* kleiner als **MQGMO_VERSION_4** ist.

ResolvedQName (MQCHAR48)

Dies ist ein Ausgabefeld, das der Warteschlangenmanager, wie vom lokalen Warteschlangenmanager definiert, auf den lokalen Namen der Warteschlange setzt, von der die Nachricht abgerufen wurde. Dieser Name weicht von dem Namen ab, der zum Öffnen der Warteschlange verwendet wurde, wenn:

- Eine Aliaswarteschlange wurde geöffnet (in diesem Fall wird der Name der lokalen Warteschlange, die den Alias aufgelöst hat, zurückgegeben) oder
- eine Modellwarteschlange wurde geöffnet (in diesem Fall wird der Name der dynamischen lokalen Warteschlange zurückgegeben).

Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ReturnedLength (MQLONG)

Dies ist ein Ausgabefeld, das der Warteschlangenmanager auf die Länge in Bytes der Nachrichtendaten setzt, die vom MQGET-Aufruf im *Buffer*-Parameter zurückgegeben wurde. Falls der Warteschlangenmanager diese Funktion nicht unterstützen sollte, wird *ReturnedLength* auf den Wert MQRL_UNDEFINED gesetzt.

Wenn Codierungen oder Zeichensätze von Nachrichten konvertiert werden, ändert sich möglicherweise die Größe der Nachrichtendaten. Bei Rückgabe durch einen MQGET-Aufruf:

- Falls *ReturnedLength nicht* MQRL_UNDEFINED ist, wird die Anzahl der Bytes der zurückgegebenen Nachrichtendaten von *ReturnedLength* angegeben.
- Falls *ReturnedLength* dem Wert MQRL_UNDEFINED entspricht, wird die Zahl der Bytes der zurückgegebenen Nachrichtendaten normalerweise entweder von *BufferLength* oder *DataLength* angegeben, und zwar von dem kleineren der beiden Werte. Der Wert kann allerdings auch *kleiner* sein und zwar, falls der MQGET-Aufruf mit dem Ursachencode MQRC_TRUNCATED_MSG_ACCEPTED abschließt. Wenn dies geschieht, werden die unkritischen Bytes im *Buffer*-Parameter auf null gesetzt.

Der folgende spezielle Wert ist definiert:

MQRL_UNDEFINED
Länge der zurückgegebenen Daten nicht definiert

Bei z/OS ist der Wert, der vom *ReturnedLength*-Feld zurückgegeben wird, immer MQRL_UNDEFINED.

Der Anfangswert dieses Felds ist MQRL_UNDEFINED. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQGMO_VERSION_3.

Segmentation (MQCHAR)

Dies ist ein Flag, das angibt, ob für die abgerufene Nachricht eine weitere Segmentierung zulässig ist. Es entspricht einem der folgenden Werte:

MQSEG_INHIBITED
Segmentierung ist nicht zulässig.

MQSEG_ALLOWED
Segmentierung zulässig

Bei z/OS setzt der Warteschlangenmanager dieses Feld immer auf MQSEG_INHIBITED.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Felds ist MQSEG_INHIBITED. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.

SegmentStatus (MQCHAR)

Dieses Flag gibt an, ob die abgerufene Nachricht ein Segment einer logischen Nachricht ist. Es entspricht einem der folgenden Werte:

MQSS_NOT_A_SEGMENT
Nachricht ist kein Segment.

MQSS_SEGMENT
Nachricht ist ein Segment, aber nicht das letzte Segment der logischen Nachricht

MQSS_LAST_SEGMENT
Nachricht ist das letzte Segment der logischen Nachricht

Dies ist auch der Wert, der zurückgegeben wird, wenn die logische Nachricht nur aus einem Segment besteht.

Bei z/OS setzt der Warteschlangenmanager dieses Feld immer auf MQSS_NOT_A_SEGMENT.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Felds ist MQSS_NOT_A_SEGMENT. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQGMO_VERSION_2 ist.

Signal1 (MQLONG)

Dies ist ein Eingabefeld, das nur zusammen mit der MQGMO_SET_SIGNAL-Option verwendet wird; es gibt ein Signal an, das übermittelt werden soll, wenn eine Nachricht verfügbar ist.

Anmerkung: Der Datentyp und die Verwendung dieses Felds werden von der Umgebung bestimmt; daher dürfen Anwendungen, die Sie zwischen verschiedenen Umgebungen portieren, keine Signale verwenden.

- Bei z/OS muss dieses Feld die Adresse eines Ereignissteuerblocks (ECB) enthalten. Der ECB muss vor Ausgabe des MQGET-Aufrufs von der Anwendung gelöscht werden. Der Speicher, in dem sich der ECB befindet, darf nicht freigegeben werden, bevor die Warteschlange nicht geschlossen ist. Der ECB wird vom Warteschlangenmanager mit einem der beschriebenen Signalbeendigungscodes gesendet. Diese Beendigungscodes werden mit den Bits 2 bis 31 des ECB angegeben, dem Bereich, der im z/OS-Zuordnungsmakro IHAECB als für einen Benutzerbeendigungscode vorgesehen gekennzeichnet ist.
- In allen anderen Umgebungen ist dies ein reserviertes Feld; sein Wert ist nicht von Bedeutung.

Folgende Signalbeendigungscodes gibt es:

MQEC_MSG_ARRIVED
In der Warteschlange wurde eine geeignete Nachricht eingereiht. Diese Nachricht wurde nicht für das aufrufende Programm reserviert. Eine zweite MQGET-Anfrage muss ausgegeben werden, aber eine andere Anwendung ruft die Nachricht eventuell ab, bevor die zweite Anfrage ausgeführt werden kann.

MQEC_WAIT_INTERVAL_EXPIRED
Das angegebene *WaitInterval* ist abgelaufen, ohne dass eine geeignete Nachricht eintraf.

MQEC_WAIT_CANCELED
Der Wartestatus wurde aus einem unbestimmten Grund beendet, beispielsweise kann der Warteschlangenmanager ihn beendet haben oder die Warteschlange wurde inaktiviert. Stellen Sie die Anfrage erneut, falls Sie eine weiterführende Diagnose wünschen.

MQEC_Q_MGR_QUIESCING
Der Wartestatus wurde beendet, da der Warteschlangenmanager in den Quiescestatus gewechselt ist (beim MQGET-Aufruf wurde MQGMO_FAIL_IF_QUIESCING angegeben).

MQEC_CONNECTION_QUIESCING
Der Wartestatus wurde abgebrochen, da die Verbindung in den Quiescestatus gewechselt ist (beim MQGET-Aufruf wurde MQGMO_FAIL_IF_QUIESCING angegeben).

Der Anfangswert dieses Felds wird von der Umgebung bestimmt:

- Bei z/OS entspricht der Anfangswert dem Nullzeiger.
- Bei allen anderen Umgebungen ist der Anfangswert `0`.

Signal2 (MQLONG)

Dies ist ein Eingabefeld, das nur zusammen mit der MQGMO_SET_SIGNAL-Option verwendet wird. Es ist ein reserviertes Feld; sein Wert ist nicht signifikant.

Der Anfangswert dieses Felds ist `0`.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQGMO_STRUC_ID
ID für die Struktur der Option zum Abrufen von Nachrichten.

Für die Programmiersprache C ist auch die Konstante MQGMO_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQGMO_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQGMO_STRUC_ID.

Version (MQLONG)
Version ist die Strukturversionsnummer.

Folgende Werte sind möglich:

MQGMO_VERSION_1
Struktur der Optionen zum Abrufen von Nachrichten Version-1.

Diese Option wird in allen Umgebungen unterstützt.

MQGMO_VERSION_2
Struktur der Optionen zum Abrufen von Nachrichten Version-2.

Diese Option wird in allen Umgebungen unterstützt.

MQGMO_VERSION_3
Struktur der Optionen zum Abrufen von Nachrichten Version-3.

Diese Option wird in allen Umgebungen unterstützt.

MQGMO_VERSION_4
Struktur der Optionen zum Abrufen von Nachrichten Version-4.

Diese Option wird in allen Umgebungen unterstützt.

Felder, die nur in den aktuelleren Versionen der Struktur vorliegen, sind in den Beschreibungen der Felder als solche angegeben. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQGMO_CURRENT_VERSION
Aktuelle Version der Nachrichtenabrufoptionsstruktur

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQGMO_VERSION_1.

WaitInterval (MQLONG)

Die näherungsweise berechnete Zeit (in Millisekunden), die der MQGET-Aufruf auf den Eingang einer Nachricht wartet, die die im Parameter *MsgDesc* des MQGET-Aufrufs angegebenen Auswahlkriterien erfüllt.

.

Wichtig: Ist sofort eine angemessene Nachricht verfügbar, kommt es zu keiner Wartezeit oder Verzögerung.

Nähere Informationen finden Sie in der Beschreibung des Feldes *MsgId* im Abschnitt „MQMD - Nachrichtendeskriptor” auf Seite 399. Wenn diese Zeitspanne verstreicht, ohne dass eine geeignete Nachricht eintrifft, schließt der Aufruf mit MQCC_FAILED und dem Ursachencode MQRC_NO_MSG_AVAILABLE ab.

Bei z/OS wird die Zeitspanne, die der MQGET-Aufruf wartet, von der Systemauslastung und der Arbeitsplanung beeinflusst und kann zwischen dem für *WaitInterval* angegebenen Wert und einem Wert ungefähr 250 Millisekunden länger als *WaitInterval* variieren.

WaitInterval wird zusammen mit der MQGMO_WAIT- oder der MQGMO_SET_SIGNAL-Option verwendet. Sie wird ignoriert, wenn keiner der beiden angegeben ist. Wird eine der Optionen angegeben, muss *WaitInterval* größer oder gleich null sein oder den folgenden Sonderwert haben:

MQWI_UNLIMITED
Unbegrenztes Warteintervall.

Der Anfangswert dieses Felds ist 0.

Anfangswerte und Sprachendeklarationen für MQGMO

Tabelle 508. Anfangswerte von Feldern in MQGMO für MQGMO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQGMO_STRUC_ID	'GMO¬'
Version	MQGMO_VERSION_1	1
Options	MQGMO_NO_WAIT	0
WaitInterval	--	0
Signal1	--	Nullzeiger bei z/OS; sonst 0
Signal2	--	0
ResolvedQName	--	Nullzeichenfolge oder Leerzeichen.
MatchOptions	MQMO_MATCH_MSG_ID und MQMO_MATCH_CORREL_ID	3
GroupStatus	MQGS_NOT_IN_GROUP	'¬'
SegmentStatus	MQSS_NOT_A_SEGMENT	'¬'
Segmentation	MQSEG_INHIBITED	'¬'
Reserved1	--	'¬'
MsgToken	MQMTOK_NONE	Nullen
ReturnedLength	MQRL_UNDEFINED	-1
Reserved2	--	'¬'
MsgHandle	MQHM_NONE	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQGMO_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of */
                             MQGET */
  MQLONG    WaitInterval;    /* Wait interval */
  MQLONG    Signal1;         /* Signal */
  MQLONG    Signal2;         /* Signal identifier */
  MQCHAR48  ResolvedQName;   /* Resolved name of destination queue */
  /* Ver:1 */
  MQLONG    MatchOptions;    /* Options controlling selection */
                             criteria used for MQGET */
  MQCHAR    GroupStatus;     /* Flag indicating whether message */
```

```
                                retrieved is in a group */
  MQCHAR    SegmentStatus;   /* Flag indicating whether message */
                                retrieved is a segment of a logical */
                                message */
  MQCHAR    Segmentation;    /* Flag indicating whether further */
                                segmentation is allowed for the */
                                message retrieved */
  MQCHAR    Reserved1;       /* Reserved */
  /* Ver:2 */
  MQBYTE16  MsgToken;        /* Message token */
  MQLONG    ReturnedLength;  /* Length of message data returned */
                                (bytes) */
  /* Ver:3 */
  MQLONG    Reserved2;       /* Reserved */
  MQHMSG    MsgHandle;       /* Message handle */
  /* Ver:4 */
};
```

- Unter z/OSwird das Feld *Signal1* als PMQLONGdeklariert.

COBOL-Delaration

```
**   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.
```

- Unter z/OSwird das Feld *Signal1* als POINTERdeklariert.

Deklaration in PL/I

```
dcl
 1 MQGMO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action of
                                      MQGET */
   3 WaitInterval   fixed bin(31), /* Wait interval */
   3 Signal1        fixed bin(31), /* Signal */
   3 Signal2        fixed bin(31), /* Signal identifier */
   3 ResolvedQName  char(48),      /* Resolved name of destination
                                      queue */
   3 MatchOptions   fixed bin(31), /* Options controlling selection
                                      criteria used for MQGET */
```

```
  3 GroupStatus    char(1),       /* Flag indicating whether message
                                     retrieved is in a group */
  3 SegmentStatus  char(1),       /* Flag indicating whether message
                                     retrieved is a segment of a logical
                                     message */
  3 Segmentation   char(1),       /* Flag indicating whether further
                                     segmentation is allowed for the
                                     message retrieved */
  3 Reserved1      char(1),       /* Reserved */
  3 MsgToken       char(16),      /* Message token */
  3 ReturnedLength fixed bin(31); /* Length of message data returned
                                     (bytes) */
  3 Reserved2      fixed bin(31); /* Reserved */
  3 MsgHandle      fixed bin(63); /* Message handle */
```

- Unter z/OSwird das Feld *Signal1* als `pointer`deklariert.

Deklaration in High Level Assembler

```
MQGMO                 DSECT
MQGMO_STRUCID         DS   CL4   Structure identifier
MQGMO_VERSION         DS   F     Structure version number
MQGMO_OPTIONS         DS   F     Options that control the action of
*                                MQGET
MQGMO_WAITINTERVAL    DS   F     Wait interval
MQGMO_SIGNAL1         DS   F     Signal
MQGMO_SIGNAL2         DS   F     Signal identifier
MQGMO_RESOLVEDQNAME   DS   CL48  Resolved name of destination queue
MQGMO_MATCHOPTIONS    DS   F     Options controlling selection criteria
*                                used for MQGET
MQGMO_GROUPSTATUS     DS   CL1   Flag indicating whether message
*                                retrieved is in a group
MQGMO_SEGMENTSTATUS   DS   CL1   Flag indicating whether message
*                                retrieved is a segment of a logical
*                                message
MQGMO_SEGMENTATION    DS   CL1   Flag indicating whether further
*                                segmentation is allowed for the message
*                                retrieved
MQGMO_RESERVED1       DS   CL1   Reserved
MQGMO_MSGTOKEN        DS   XL16  Message token
MQGMO_RETURNEDLENGTH  DS   F     Length of message data returned (bytes)
MQGMO_RESERVED2       DS   F     Reserved
MQGMO_MSGHANDLE       DS   D     Message handle
MQGMO_LENGTH          EQU  *-MQGMO
                      ORG  MQGMO
MQGMO_AREA            DS   CL(MQGMO_LENGTH)
```

Deklaration in Visual Basic

```
Type MQGMO
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Options        As Long      'Options that control the action of MQGET'
  WaitInterval   As Long      'Wait interval'
  Signal1        As Long      'Signal'
  Signal2        As Long      'Signal identifier'
  ResolvedQName  As String*48 'Resolved name of destination queue'
  MatchOptions   As Long      'Options controlling selection criteria'
                              'used for MQGET'
  GroupStatus    As String*1  'Flag indicating whether message'
                              'retrieved is in a group'
  SegmentStatus  As String*1  'Flag indicating whether message'
                              'retrieved is a segment of a logical'
                              'message'
  Segmentation   As String*1  'Flag indicating whether further'
                              'segmentation is allowed for the message'
                              'retrieved'
  Reserved1      As String*1  'Reserved'
  MsgToken       As MQBYTE16  'Message token'
  ReturnedLength As Long      'Length of message data returned (bytes)'
End Type
```

MQIIH - IMS-Header

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 509. Felder im MQIIH		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
StrucLength	Länge der MQIIH-Struktur	StrucLength
Encoding	Reserved	Encoding
CodedCharSetId	Reserved	CodedCharSetId
Format	Name des MQ-Formats von Daten hinter dem MQIIH	Format
Flags	Markierungen	Flags
LTermOverride	Überschreibung des logischen Terminals	LTermOverride
MFSMapName	Mapname für Nachrichtenformatservices	MFSMapName
ReplyToFormat	Name des MQ-Formats der Antwortnachricht	ReplyToFormat
Authenticator	RACF™-Kennwort oder -Passticket	Authenticator
TranInstanceId	Transaktionsinstanz-ID	TranInstanceId
TranState	Transaktionsstatus	TranState
CommitMode	Festschreibungsmodus	CommitMode
SecurityScope	Sicherheitsbereich	SecurityScope
Reserved	Reserved	Reserved

Überblick für MQIIH

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Die MQIIH-Struktur beschreibt die Informationen, die am Anfang einer Nachricht vorhanden sein müssen, die über WebSphere MQ for z/OSan die IMS -Brücke gesendet wird.

Formatname: MQFMT_IMS.

Zeichensatz und Codierung: Für den Zeichensatz und die Codierung, die für die MQIIH-Struktur und Anwendungsnachrichtendaten verwendet werden, gelten besondere Bedingungen:

- Anwendungen, die Verbindungen mit dem Warteschlangenmanager herstellen, der Eigner der IMS-Brücken-WS ist, müssen eine MQIIH-Struktur mit Zeichensatz und Codierung des Warteschlangenmanagers bereitstellen. Der Grund hierfür ist, dass die Datenkonvertierung der MQIIH-Struktur in diesem Fall nicht ausgeführt wird.
- Anwendungen, die eine Verbindung zu anderen Warteschlangenmanagern herstellen, können eine MQIIH-Struktur mit jedem der unterstützten Zeichensätze und jeder der unterstützten Codierungen bereitstellen; der empfangende Nachrichtenkanalagent, der mit dem Warteschlangenmanager verbunden ist, der der Eigner der IMS-Brückenwarteschlange ist, wandelt den MQIIH um.
- Die Anwendungsnachrichtendaten, die der MQIIH-Struktur folgen, müssen denselben Zeichensatz und dieselbe Codierung aufweisen wie die MQIIH-Struktur. Verwenden Sie nicht das *CodedCharSetId*- und das *Encoding*-Feld in der MQIIH-Struktur, um den Zeichensatz und die Codierung der Anwendungsnachrichtendaten anzugeben.

Sie müssen einen Datenkonvertierungsexit bereitstellen, um die Anwendungsnachrichtendaten zu konvertieren, wenn die Daten nicht einem der integrierten Formate entsprechen, die vom Warteschlangenmanager unterstützt werden.

Felder für MQIIH

Die MQIIH-Struktur enthält die nachfolgend aufgeführten Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

Authenticator (MQCHAR8)

Dies ist das RACF-Kennwort oder -PassTicket. Es ist optional; wenn es angegeben wird, wird es zusammen mit der Benutzer-ID im MQMD-Sicherheitskontext verwendet, um ein Benutzertoken zu erstellen, das an IMS zur Bereitstellung eines Sicherheitskontexts gesendet wird. Wenn es nicht angegeben wird, wird die Benutzer-ID ohne Bestätigung verwendet. Dies hängt von der Einstellung der RACF-Schalter ab, die gegebenenfalls einen Authentifikator erfordern.

Das wird ignoriert, wenn das erste Byte leer oder null ist. Folgende Sonderwerte sind zulässig:

MQIAUT_NONE
Keine Authentifizierung.

Für die Programmiersprache C ist auch die Konstante MQIAUT_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQIAUT_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge des Felds wird durch MQ_AUTHENTICATOR_LENGTH angegeben. Der Anfangswert dieses Felds ist MQIAUT_NONE.

CodedCharSetId (MQLONG)

Dies ist ein reserviertes Feld, dessen Wert nicht von Bedeutung ist. Der Anfangswert dieses Feldes ist 0.

Die Zeichensatz-ID für Unterstützungsstrukturen, die auf eine MQIIH-Struktur folgen, ist mit der Zeichensatz-ID der MQIIH-Struktur identisch und wird einem führenden MQ-Header entnommen.

CommitMode (MQCHAR)

Dies ist der IMS-Commitmodus. Weitere Informationen zu IMS -Commitmodi finden Sie in der *OTMA-Referenz* . Folgende Werte sind möglich:

MQICM_COMMIT_THEN_SEND
Commit durchführen, dann senden.

Dieser Modus schließt die doppelte Warteschlangensteuerung der Ausgabe ein, jedoch kürzere Bereichsbelegungszeiten. Direktaufruf- und Dialogtransaktionen können in diesem Modus nicht ausgeführt werden.

MQICM_SEND_THEN_COMMIT
Senden, dann Commit durchführen.

Jede IMS-Transaktion, die aufgrund eines Commitmodus von MQICM_SEND_THEN_COMMIT eingeleitet wird, wird im RESPONSE-Modus ausgeführt, unabhängig davon, wie die Transaktion in der IMS-Systemdefinition definiert ist (MSGTYPE-Parameter im TRANSACT-Makro). Dies gilt auch für Transaktionen, die aufgrund eines Transaktionswechsels ausgelöst wurden.

Der Anfangswert dieses Felds ist MQICM_COMMIT_THEN_SEND.

Encoding (MQLONG)

Dies ist ein reserviertes Feld, dessen Wert nicht von Bedeutung ist. Der Anfangswert dieses Feldes ist 0.

Die Codierung für Unterstützungsstrukturen, die auf eine MQIIH-Struktur folgen, ist mit der Codierung der MQIIH-Struktur identisch und wird einem führenden MQ-Header entnommen.

Flags (MQLONG)

Der Flagwert muss wie folgt lauten:

MQIIH_NONE
Keine Flags.

MQIIH_PASS_EXPIRATION
Die Antwortnachricht enthält:

- Dieselben Ablaufberichtsoptionen wie die Anforderungsnachricht.
- Die verbleibende Ablaufzeit der Anforderungsnachricht ohne Anpassung für die Verarbeitungszeit der Bridge.

Wird dieser Wert nicht gesetzt, wird die Ablaufzeit auf *unbegrenzt* gesetzt.

MQIIH_REPLY_FORMAT_NONE
Setzt das MQIIH.Format-Feld der Antwort auf MQFMT_NONE.

MQIIH_IGNORE_PURG
Gibt den TMAMIPRG-Anzeiger im OTMA-Präfix an, der anfordert, dass OTMA PURG-Aufrufe auf den Telekommunikationsprogramm-PCB für CM0-Transaktionen ignoriert.

MQIIH_CM0_REQUEST_RESPONSE
Bei CM0-Transaktionen ("Commit Mode 0") setzt dieses Flag den TMAMHRSP-Anzeiger im OTMA-Präfix. Wird dieser Anzeiger gesetzt, so wird angefordert, dass OTMA/IMS eine "DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY"-Nachricht erstellen soll, wenn das ursprüngliche IMS-Anwendungsprogramm weder auf den Eingabe-/Ausgabe-PCB antwortet, noch die Nachricht zu einer anderen Transaktion wechselt.

Der Anfangswert dieses Felds ist MQIIH_NONE.

Format (MQCHAR8)

In diesem Feld wird der MQ-Formatname der Daten angegeben, die der MQIIH-Struktur folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

LTermOverride (MQCHAR8)

Der Korrekturwert des logischen Terminals, der im I/O-PCB-Feld angeordnet ist. Er ist optional; wird er nicht angegeben, wird der TPIPE-Name verwendet. Er wird ignoriert, wenn das erste Byte leer oder null ist.

Die Länge dieses Felds wird durch MQ_LTERM_OVERRIDE_LENGTH angegeben. Der Anfangswert dieses Felds ist 8 Leerzeichen.

MFSMapName (MQCHAR8)

Der Mapname der Nachrichtenformatservices, der im I/O-PCB-Feld angeordnet ist. Er ist optional. Bei der Eingabe stellt er den Nachrichteneingabedeskriptor dar, bei der Ausgabe den Nachrichtenausgabedeskriptor. Es wird ignoriert, falls das erste Byte leer oder null ist.

Die Länge dieses Felds wird durch MQ_MFS_MAP_NAME_LENGTH angegeben. Der Anfangswert dieses Felds ist 8 Leerzeichen.

ReplyToFormat (MQCHAR8)

Dies ist der MQ-Formatname der Antwortnachricht, die als Antwort auf die aktuelle Nachricht gesendet wird. Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

Geben Sie, um die Daten der Antwortnachricht mithilfe von MQGMO_CONVERT umzuwandeln, entweder MQIIH.replyToFormat=MQFMT_STRING oder MQIIH.replyToFormat=MQFMT_IMS_VAR_STRING an. Eine

Erläuterung zur Verwendung dieser Felder finden Sie im Abschnitt über „Format (MQCHAR8)” auf Seite 414.

Wenn bei der Anforderungsnachricht der Standardwert MQIIH.replyToFormat=MQFMT_NONE verwendet wird und die Antwortnachricht unter Verwendung von MQGMO_CONVERT abgerufen wird, wird keine Datenkonvertierung durchgeführt.

Reserved (MQCHAR)

Dies ist ein reserviertes Feld; es muss leer sein.

SecurityScope (MQCHAR)

Dies gibt die erforderliche IMS-Sicherheitsverarbeitung an. Die folgenden Werte sind definiert:

MQISS_CHECK
Sicherheitsbereich überprüfen: Ein ACEE ist in der Steuerregion integriert, aber nicht in der abhängigen Region.

MQISS_FULL
Vollständiger Sicherheitsbereich: Ein zwischengespeichertes ACEE ist in der Steuerregion und ein nicht-zwischengespeichertes ACEE ist in die abhängige Region integriert. Wenn Sie MQISS_FULL verwenden, stellen Sie sicher, dass die Benutzer-ID, für die das ACEE erstellt wird, Zugriff auf die in der abhängigen Region verwendeten Ressourcen hat.

Wenn für dieses Feld weder MQISS_CHECK noch MQISS_FULL angegeben ist, wird MQISS_CHECK vorausgesetzt.

Der Anfangswert dieses Felds ist MQISS_CHECK.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQIIH_STRUC_ID
ID für die IMS-Headerstruktur.

Für die Programmiersprache C ist auch die Konstante MQIIH_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQIIH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQIIH_STRUC_ID.

StrucLength (MQLONG)

Dies ist die Länge der MQIIH-Struktur. Folgende Werte sind möglich:

MQIIH_LENGTH_1
Länge der IMS-Headerstruktur.

Der Anfangswert dieses Felds ist MQIIH_LENGTH_1.

TranInstanceId (MQBYTE16)

Dies ist die Transaktionsinstanz-ID. Dieses Feld wird von Ausgabenachrichten von IMS verwendet, wird also bei der ersten Eingabe ignoriert. Wenn Sie *TranState* auf MQITS_IN_CONVERSATION setzen, muss diese Information bei der nächsten und allen nachfolgenden Eingaben bereitgestellt werden, um IMS zu ermöglichen, die Nachrichten mit den richtigen Dialogen zu korrelieren. Sie können den folgenden Spezialwert verwenden:

MQITII_NONE
Keine Transaktionsinstanz-ID.

Für die Programmiersprache C ist auch die Konstante MQITII_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQITII_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_TRAN_INSTANCE_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQITII_NONE.

TranState (MQCHAR)

Dieses Feld gibt den IMS-Dialogstatus an. Bei der ersten Eingabe wird es ignoriert, weil kein Dialog existiert. Bei nachfolgenden Eingaben gibt es an, ob ein Dialog aktiv ist oder nicht. Bei der Ausgabe wird es von IMS gesetzt. Folgende Werte sind möglich:

MQITS_IN_CONVERSATION
Im Dialog.

MQITS_NOT_IN_CONVERSATION
Nicht im Dialog.

MQITS_ARCHITECTED
Zustandsdaten der Transaktion werden in gestalteter Form zurückgegeben.

Dieser Wert wird nur mit dem Befehl IMS `/DISPLAY TRAN` verwendet. Er gibt die Zustandsdaten der Transaktion in IMS-gestalteter Form statt in Zeichenform zurück.

Der Anfangswert ist MQITS_NOT_IN_CONVERSATION.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQIIH_VERSION_1
Versionsnummer der IMS-Headerstruktur

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQIIH_CURRENT_VERSION
Aktuelle Version der IMS-Headerstruktur

Der Anfangswert dieses Felds ist MQIIH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQIIH

Tabelle 510. Anfangswerte von Feldern in MQIIH für MQIIH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQIIH_STRUC_ID	`'IIH?'`
Version	MQIIH_VERSION_1	1
StrucLength	MQIIH_LENGTH_1	84
Encoding	--	0
CodedCharSetId	--	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQIIH_NONE	0
LTermOverride	--	Leerzeichen
MFSMapName	--	Leerzeichen
ReplyToFormat	MQFMT_NONE	Leerzeichen
Authenticator	MQIAUT_NONE	Leerzeichen
TranInstanceId	MQITII_NONE	Nullen
TranState	MQITS_NOT_IN_CONVERSATION	`'?'`
CommitMode	MQICM_COMMIT_THEN_SEND	`'0'`

Tabelle 510. Anfangswerte von Feldern in MQIIH für MQIIH (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
SecurityScope	MQISS_CHECK	'C'
Reserved	--	'?'

Anmerkungen:

1. Das Symbol ? steht für ein einzelnes Leerzeichen.
2. In der Programmiersprache C enthält die MakrovariableMQIIH_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQIIH structure */
  MQLONG    Encoding;        /* Reserved */
  MQLONG    CodedCharSetId;  /* Reserved */
  MQCHAR8   Format;          /* MQ format name of data that follows
                                MQIIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR8   LTermOverride;   /* Logical terminal override */
  MQCHAR8   MFSMapName;      /* Message format services map name */
  MQCHAR8   ReplyToFormat;   /* MQ format name of reply message */
  MQCHAR8   Authenticator;   /* RACF password or passticket */
  MQBYTE16  TranInstanceId;  /* Transaction instance identifier */
  MQCHAR    TranState;       /* Transaction state */
  MQCHAR    CommitMode;      /* Commit mode */
  MQCHAR    SecurityScope;   /* Security scope */
  MQCHAR    Reserved;        /* Reserved */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQIIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQIIH structure */
  3 Encoding       fixed bin(31), /* Reserved */
  3 CodedCharSetId fixed bin(31), /* Reserved */
  3 Format         char(8),       /* MQ format name of data that follows
                                     MQIIH */
  3 Flags          fixed bin(31), /* Flags */
  3 LTermOverride  char(8),       /* Logical terminal override */
  3 MFSMapName     char(8),       /* Message format services map name */
  3 ReplyToFormat  char(8),       /* MQ format name of reply message */
  3 Authenticator  char(8),       /* RACF password or passticket */
  3 TranInstanceId char(16),      /* Transaction instance identifier */
  3 TranState      char(1),       /* Transaction state */
  3 CommitMode     char(1),       /* Commit mode */
  3 SecurityScope  char(1),       /* Security scope */
  3 Reserved       char(1);       /* Reserved */
```

Deklaration in High Level Assembler

```
MQIIH                 DSECT
MQIIH_STRUCID         DS   CL4   Structure identifier
MQIIH_VERSION         DS   F     Structure version number
MQIIH_STRUCLENGTH     DS   F     Length of MQIIH structure
MQIIH_ENCODING        DS   F     Reserved
MQIIH_CODEDCHARSETID  DS   F     Reserved
MQIIH_FORMAT          DS   CL8   MQ format name of data that follows
*                                MQIIH
MQIIH_FLAGS           DS   F     Flags
MQIIH_LTERMOVERRIDE   DS   CL8   Logical terminal override
MQIIH_MFSMAPNAME      DS   CL8   Message format services map name
MQIIH_REPLYTOFORMAT   DS   CL8   MQ format name of reply message
MQIIH_AUTHENTICATOR   DS   CL8   RACF password or passticket
MQIIH_TRANINSTANCEID  DS   XL16  Transaction instance identifier
MQIIH_TRANSTATE       DS   CL1   Transaction state
MQIIH_COMMITMODE      DS   CL1   Commit mode
MQIIH_SECURITYSCOPE   DS   CL1   Security scope
MQIIH_RESERVED        DS   CL1   Reserved
*
MQIIH_LENGTH          EQU  *-MQIIH
                      ORG  MQIIH
MQIIH_AREA            DS   CL(MQIIH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQIIH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQIIH structure'
  Encoding       As Long     'Reserved'
  CodedCharSetId As Long     'Reserved'
  Format         As String*8 'MQ format name of data that follows MQIIH'
  Flags          As Long     'Flags'
  LTermOverride  As String*8 'Logical terminal override'
  MFSMapName     As String*8 'Message format services map name'
  ReplyToFormat  As String*8 'MQ format name of reply message'
  Authenticator  As String*8 'RACF password or passticket'
  TranInstanceId As MQBYTE16 'Transaction instance identifier'
  TranState      As String*1 'Transaction state'
  CommitMode     As String*1 'Commit mode'
  SecurityScope  As String*1 'Security scope'
```

```
  Reserved       As String*1 'Reserved'
End Type
```

MQIMPO – Optionen für das Abfragen von Nachrichteneigenschaften

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. MQIMPO-Struktur – Optionen für die Abfrage von Nachrichteneigenschaften

Tabelle 511. Felder in MQIMPO		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQINQMP	Options
RequestedEncoding	Codierung, in die die abgefragte Eigenschaft umgewandelt werden soll	RequestedEncoding
RequestedCCSID	Zeichensatz der abgefragten Eigenschaft	RequestedCCSID
ReturnedEncoding	Codierung des zurückgegebenen Werts	ReturnedEncoding
ReturnedCCSID	Zeichensatz des zurückgegebenen Werts	ReturnedCCSID
Reserved1	Reserviertes Feld	ReturnedCCSID
ReturnedName	Name der abgefragten Eigenschaft	ReturnedName
TypeString	Zeichenfolgedarstellung des Datentyps der Eigenschaft	TypeString

Überblick über MQIMPO

Die Optionsstruktur zur Abfrage der Nachrichteneigenschaften.

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Mit der MQIMPO-Struktur können Anwendungen Optionen angeben, die steuern, wie die Eigenschaften von Nachrichten abgefragt werden. Die Struktur ist ein Eingabeparameter für den MQINQMP-Aufruf.

Zeichensatz und Codierung: Die Daten in MQIMPO müssen im Zeichensatz und in der Codierung der Anwendung (MQENC_NATIVE) enthalten sein.

Felder für MQIMPO

Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Felder

Die MQIMPO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld Options

Die folgenden Optionen steuern das Verhalten von MQINQMP. Sie können eine oder mehrere dieser Optionen angeben; wenn Sie mehr als eine benötigen, können die Werte:

- gemeinsam hinzugefügt werden (dieselbe Konstante nicht mehr als einmal hinzufügen) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

Auf ungültige Kombinationen von Optionen wird hingewiesen, alle anderen Kombinationen sind gültig.

Optionen für Wertedaten: Die folgenden Optionen beziehen sich auf die Verarbeitung der Wertedaten, wenn die Eigenschaft aus der Nachricht abgerufen wird.

MQIMPO_CONVERT_VALUE
Diese Option fordert an, den Wert der Eigenschaft umzuwandeln, sodass sie den angegebenen Werten *RequestedCCSID* und *RequestedEncoding* entspricht, bevor der MQINQMP-Aufruf den Eigenschaftswert im Bereich *Value* zurückgibt.

- Wenn die Konvertierung erfolgreich ist, werden die Felder *ReturnedCCSID* und *ReturnedEncoding* bei der Rückgabe des MQINQMP-Aufrufs auf *RequestedCCSID* und *RequestedEncoding* gesetzt.
- Schlägt die Umwandlung fehl, aber der MQINQMP-Aufruf wird ansonsten ohne Fehler abgeschlossen, wird der Eigenschaftswert ohne Umwandlung zurückgegeben.

  Ist die Eigenschaft eine Zeichenfolge, werden die Felder *ReturnedCCSID* und *ReturnedEncoding* auf den Zeichensatz und die Codierung der nicht umgewandelten Zeichenfolge gesetzt.

  Der Beendigungscode ist in diesem Fall MQCC_WARNING, der Ursachencode MQRC_PROP_VALUE_NOT_CONVERTED. Der Eigenschaftscursor wird auf die zurückgegebene Eigenschaft vorgesetzt.

Wird der Eigenschaftswert während der Umwandlung erweitert, sodass er die Größe des Parameters *Value* überschreitet, wird der Wert ohne Umwandlung mit dem Beendigungscode MQCC_FAILED zurückgegeben; der Ursachencode wird auf MQRC_PROPERTY_VALUE_TOO_BIG gesetzt.

Der Parameter *DataLength* des MQINQMP-Aufrufs gibt die Länge zurück, in die der Eigenschaftswert konvertiert worden wäre, damit die Anwendung die Größe des Puffers bestimmen kann, der zur Aufnahme des konvertierten Eigenschaftswerts erforderlich ist. Der Eigenschaftscursor bleibt unverändert.

Bei dieser Option muss ferner, für den Fall, dass:

- der Eigenschaftsname einen Platzhalter enthält und
- das Feld *ReturnedName* mit einer Adresse oder relativer Adresse für den zurückgegebenen Namen initialisiert ist,

In diesem Fall wird der zurückgegebene Name so konvertiert, dass er den Werten *RequestedCCSID* und *RequestedEncoding* entspricht.

- Wenn die Konvertierung erfolgreich ist, werden das Feld *VSCCSID* von *ReturnedName* und die Codierung des zurückgegebenen Namens auf den Eingabewert *RequestedCCSID* und *RequestedEncoding* gesetzt.
- Schlägt die Konvertierung fehl, wird der MQINQMP-Aufruf ansonsten aber ohne Fehler oder Warnung ausgeführt, bleibt der zurückgegebene Name unkonvertiert. Der Beendigungscode ist in diesem Fall MQCC_WARNING, der Ursachencode MQRC_PROP_NAME_NOT_CONVERTED.

  Der Eigenschaftscursor wird auf die zurückgegebene Eigenschaft vorgesetzt. MQRC_PROP_VALUE_NOT_CONVERTED wird zurückgegeben, wenn weder der Wert noch der Name umgewandelt werden.

Wird der zurückgegebene Wert während der Umwandlung erweitert, sodass er die Größe des Feldes *VSBufsize* von *RequestedName* überschreitet, wird die zurückgegebene Zeichenfolge nicht umgewandelt, es wird der Fertigungsstellungscode MQCC_FAILED und der Ursachencode MQRC_PROPERTY_NAME_TOO_BIG gesetzt.

Das Feld *VSLength* der MQCHARV-Struktur gibt die Länge zurück, in die der Eigenschaftswert konvertiert worden wäre, damit die Anwendung die Größe des Puffers ermitteln kann, der für den konvertierten Eigenschaftswert erforderlich ist. Der Eigenschaftscursor bleibt unverändert.

MQIMPO_CONVERT_TYPE
Diese Option fordert an, dass der Wert der Eigenschaft vom aktuellen Datentyp in den Datentyp konvertiert wird, der im Parameter *Type* des MQINQMP-Aufrufs angegeben ist.

- Bei erfolgreicher Umwandlung bleibt der Parameter *Type* bei Rückgabe des MQINQMP-Aufrufs unverändert.
- Schlägt die Umwandlung fehl, aber der MQINQMP-Aufruf wird ansonsten ohne Fehler abgeschlossen, schlägt der Aufruf mit dem Ursachencode MQRC_PROP_CONV_NOT_SUPPORTED fehl. Der Eigenschaftscursor bleibt unverändert.

Wenn die Umwandlung des Datentyps dazu führt, dass der Wert erweitert wird und der umgewandelte Wert die Größe des Parameters *Value* überschreitet, wird der Wert ohne Umwandlung mit dem Beendigungscode MQCC_FAILED zurückgegeben und der Ursachencode wird auf MQRC_PROPERTY_VALUE_TOO_BIG gesetzt.

Der Parameter *DataLength* des MQINQMP-Aufrufs gibt die Länge zurück, in die der Eigenschaftswert konvertiert worden wäre, damit die Anwendung die Größe des Puffers bestimmen kann, der zur Aufnahme des konvertierten Eigenschaftswerts erforderlich ist. Der Eigenschaftscursor bleibt unverändert.

Wenn der Wert des Parameters *Type* des MQINQMP-Aufrufs ungültig ist, schlägt der Aufruf mit dem Grund MQRC_PROPERTY_TYPE_ERROR fehl.

Wenn die angeforderte Datentypumwandlung nicht unterstützt wird, schlägt der Aufruf mit dem Grund MQRC_PROP_CONV_NOT_SUPPORTED fehl. Folgende Datentypumwandlungen werden unterstützt:

Datentyp der Eigenschaft	**Unterstützte Zieldatentypen**
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	--

Für die unterstützten Konvertierungen gelten folgende allgemeine Regeln:

- Numerische Eigenschaftswerte können von einem Datentyp in einen anderen umgewandelt werden, sofern bei der Umwandlung keine Daten verloren gehen.

  Zum Beispiel kann der Wert einer Eigenschaft mit dem Datentyp MQTYPE_INT32 in einen Wert mit dem Datentyp MQTYPE_INT64 umgewandelt werden, aber nicht in einen Wert mit dem Datentyp MQTYPE_INT16.
- Ein Eigenschaftswert eines Datentyps kann in eine Zeichenfolge umgewandelt werden.
- Ein Zeichenfolgen-Eigenschaftswert kann in jeden anderen Datentyp konvertiert werden, vorausgesetzt, die Zeichenfolge wird für die Konvertierung korrekt formatiert. Versucht eine Anwendung, einen nicht ordnungsgemäß formatierten Zeichenfolgeeigenschaftswert umzuwandeln, gibt WebSphere MQ den Ursachencode MQRC_PROP_NUMBER_FORMAT_ERROR zurück.
- Versucht eine Anwendung eine nicht unterstützte Umwandlung, gibt WebSphere MQ den Ursachencode MQRC_PROP_CONV_NOT_SUPPORTED zurück.

Für die Konvertierung eines Eigenschaftswerts von einem Datentyp in einen anderen gelten folgende besondere Regeln:

- Bei der Umwandlung eines MQTYPE_BOOLEAN-Eigenschaftswerts in eine Zeichenfolge wird der Wert TRUE in die Zeichenfolge "TRUE" und der Wert false in die Zeichenfolge "FALSE" umgewandelt.
- Bei der Umwandlung eines MQTYPE_BOOLEAN-Eigenschaftswerts in einen numerischen Datentyp wird der Wert TRUE in eine Eins, der Wert FALSE in eine Null umgewandelt.
- Bei der Umwandlung eines Zeichenfolgeeigenschaftswerts in einen MQTYPE_BOOLEAN-Wert wird die Zeichenfolge "TRUE" oder "1" in TRUE und die Zeichenfolge "FALSE" oder "0" in FALSE umgewandelt.

  Bei den Bedingungen "WAHR" und "FALSCH" wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Andere Zeichenfolgen können nicht umgewandelt werden; WebSphere MQ gibt den Ursachencode MQRC_PROP_NUMBER_FORMAT_ERROR zurück.
- Bei der Umwandlung eines Zeichenfolgeeigenschaftswerts in einen Wert vom Datentyp MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 oder MQTYPE_INT64 muss die Zeichenfolge das folgende Format haben:

  ```
  [blanks][sign]digits
  ```

  Die Zeichenfolge hat folgende Komponenten:

  blanks
  Optionale führende Leerzeichen

  sign
  Ein optionales Pluszeichen (+) oder Minuszeichen (-).

  digits
  Eine zusammenhängende Folge von Ziffern (0-9). Es muss mindestens ein Ziffernzeichen vorhanden sein.

  Auf die Ziffernfolge können weitere Zeichen folgen, bei denen es sich nicht um Ziffern handelt, doch wird die Konvertierung beendet, sowie das erste dieser Zeichen erreicht wird. Es wird vorausgesetzt, dass die Zeichenfolge eine Ganzzahl im Dezimalformat darstellt.

  WebSphere MQ gibt den Ursachencode MQRC_PROP_NUMBER_FORMAT_ERROR zurück, wenn die Zeichenfolge nicht ordnungsgemäß formatiert ist.
- Bei der Umwandlung eines Zeichenfolgeeigenschaftswerts mit dem Datentyp MQTYPE_FLOAT32 oder MQTYPE_FLOAT64 muss die Zeichenfolge das folgende Format haben:

  ```
      [blanks][sign]digits[.digits][e_char[e_sign]e_digits]
  ```

  Die Zeichenfolge hat folgende Komponenten:

  blanks
  Optionale führende Leerzeichen

  sign
  Ein optionales Pluszeichen (+) oder Minuszeichen (-).

  digits
  Eine zusammenhängende Folge von Ziffern (0-9). Es muss mindestens ein Ziffernzeichen vorhanden sein.

  e_char
  Ein Exponentenzeichen, das entweder "E" oder "e" ist.

  e_sign
  Ein optionales Pluszeichen (+) oder Minuszeichen (-) für den Exponenten.

e_digits
Eine zusammenhängende Folge von Ziffern (0-9) für den Exponenten. Mindestens eine Ziffer muss vorhanden sein, wenn die Zeichenfolge ein Exponentenzeichen enthält.

Auf die Ziffernfolge bzw. die optionalen, einen Exponenten darstellenden Zeichen, können weitere Zeichen folgen, bei denen es sich nicht um Ziffern handelt, doch wird die Konvertierung beendet, sowie das erste dieser Zeichen erreicht wird. Es wird davon ausgegangen, dass die Zeichenfolge eine Gleitkommazahl mit einem Exponenten der Potenz 10 darstellt.

WebSphere MQ gibt den Ursachencode MQRC_PROP_NUMBER_FORMAT_ERROR zurück, wenn die Zeichenfolge nicht ordnungsgemäß formatiert ist.

- Wird ein numerischer Eigenschaftswert in eine Zeichenfolge konvertiert, wird der Wert in dessen Zeichenfolgedarstellung als Dezimalzahl konvertiert und nicht in die Zeichenfolge, die das ASCII-Zeichen für diesen Wert enthält. So wird beispielsweise die Ganzzahl 65 in die Zeichenfolge "65" und nicht in die Zeichenfolge "A" konvertiert.
- Wird ein Bytefolgen-Eigenschaftswert in eine Zeichenfolge konvertiert, wird jedes Byte in die beiden Hexadezimalzeichen konvertiert, von denen es dargestellt wird. So wird beispielsweise das Byte-Array {0xF1, 0x12, 0x00, 0xFF} in die Zeichenfolge "F11200FF" konvertiert.

MQIMPO_QUERY_LENGTH
Fragt den Typ und die Länge des Eigenschaftswerts ab. Die Länge wird im Parameter *DataLength* des MQINQMP-Aufrufs zurückgegeben. Der Eigenschaftswert wird nicht zurückgegeben.

Wenn ein *ReturnedName*-Puffer angegeben ist, wird das Feld *VSLength* der Struktur MQCHARV mit der Länge des Eigenschaftsnamens gefüllt. Der Eigenschaftsname wird nicht zurückgegeben.

Iterationsoptionen: Die folgenden Optionen gelten für die Iteration von Eigenschaften unter Verwendung eines Namens mit einem Platzhalterzeichen.

MQIMPO_INQ_FIRST
Fragt die erste mit dem angegebenen Namen übereinstimmende Eigenschaft ab. Nach diesem Aufruf wird in der zurückgegebenen Eigenschaft ein Cursor eingerichtet.

Dies ist der Standardwert.

Anschließend kann bei Bedarf die Option MQIMPO_INQ_PROP_UNDER_CURSOR mit einem MQINQMP-Aufruf verwendet werden, um dieselbe Eigenschaft erneut abzufragen.

Da es nur einen Eigenschaftscursor gibt, wird der Cursor zurückgesetzt, wenn sich der im MQINQMP-Aufruf angegebene Eigenschaftsname ändert.

Diese Option ist mit den folgenden Optionen nicht zulässig:

MQIMPO_INQ_NEXT
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT
Fragt die nächste mit dem angegebenen Namen übereinstimmende Eigenschaft ab, wobei die Suche ab dem Eigenschaftscursor fortgesetzt wird. Der Cursor wird auf die zurückgegebene Eigenschaft gesetzt.

Ist dies der erste MQINQMP-Aufruf für den angegebenen Namen, wird die erste mit dem angegebenen Namen übereinstimmende Eigenschaft zurückgegeben.

Anschließend kann bei Bedarf die Option MQIMPO_INQ_PROP_UNDER_CURSOR mit einem MQINQMP-Aufruf verwendet werden, um dieselbe Eigenschaft erneut abzufragen.

Wurde die Eigenschaft unter dem Cursor gelöscht, gibt MQINQMP die nächste übereinstimmende Eigenschaft hinter der gelöschten zurück.

Wird eine mit dem Platzhalter übereinstimmende Eigenschaft hinzugefügt, kann sie während einer laufenden Iteration zurückgegeben werden oder auch nicht. Die Eigenschaft wird zurückgegeben, sobald die Iteration erneut mit MQIMPO_INQ_FIRST beginnt.

Eine Eigenschaft, die mit dem Platzhalter übereinstimmt, der während der laufenden Iteration gelöscht wurde, wird nach der Löschung nicht zurückgegeben.

Diese Option ist mit den folgenden Optionen nicht zulässig:

MQIMPO_INQ_FIRST
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR
Ruft den Wert der Eigenschaft ab, auf die der Eigenschaftscursor zeigt. Die Eigenschaft, auf die der Eigenschaftscursor verweist, ist die zuletzt über die Option MQIMPO_INQ_FIRST oder MQIMPO_INQ_NEXT abgefragte Eigenschaft.

Der Eigenschaftscursor wird zurückgesetzt, wenn die Nachrichtenkennung wiederverwendet wird, wenn die Nachrichtenkennung im Feld *MsgHandle* der MQGMO bei einem MQGET-Aufruf angegeben wird oder wenn die Nachrichtenkennung in *OriginalMsgHandle* -oder *NewMsgHandle* -Feldern der MQPMO-Struktur bei einem MQPUT-Aufruf angegeben wird.

Wird diese Option verwendet, wenn noch kein Eigenschaftscursor eingerichtet wurde oder wenn die Eigenschaft, auf die der Eigenschaftscursor verweist, gelöscht wurde, schlägt der Aufruf mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl.

Diese Option ist mit den folgenden Optionen nicht zulässig:

MQIMPO_INQ_FIRST
MQIMPO_INQ_NEXT

Wenn keine der zuvor beschriebenen Optionen erforderlich ist, kann die folgende Option verwendet werden:

MQIMPO_NONE
Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an.

MQIMPO_NONE unterstützt die Programmdokumentation. Diese Option soll mit keiner anderen Option verwendet werden, aber da ihr Wert Null ist, kann ihre Verwendung nicht erkannt werden.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Feldes ist MQIMPO_INQ_FIRST.

RequestedCCSID (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld RequestedCCSID

Der Zeichensatz, in den der abgefragte Eigenschaftswert konvertiert werden soll, wenn der Wert eine Zeichenfolge ist. Dies ist auch der Zeichensatz, in den der *ReturnedName* konvertiert werden soll, wenn MQIMPO_CONVERT_VALUE oder MQIMPO_CONVERT_TYPE angegeben ist.

Der Anfangswert dieses Felds ist MQCCSI_APPL.

RequestedEncoding (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld RequestedEncoding

Dies ist die Codierung, in die der abgefragte Eigenschaftswert umgewandelt wird, wenn MQIMPO_CONVERT_VALUE oder MQIMPO_CONVERT_TYPE angegeben ist.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

Reserved1 (MQCHAR)

Dies ist ein reserviertes Feld. Der Anfangswert dieses Felds ist ein Leerzeichen (4-Byte-Feld).

ReturnedCCSID (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld ReturnedCCSID

Bei der Ausgabe ist dies der Zeichensatz des Werts, der zurückgemeldet wird, wenn der Parameter *Type* im MQINQMP-Aufruf MQTYPE_STRING ist.

Wenn die Option MQIMPO_CONVERT_VALUE angegeben ist und die Konvertierung erfolgreich war, hat das Feld *ReturnedCCSID* bei der Rückgabe denselben Wert wie der eingegebene Wert.

Der Anfangswert dieses Felds ist null.

ReturnedEncoding (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld ReturnedEncoding

Bei der Ausgabe ist dies die Codierung des zurückgegebenen Werts.

Wenn die Option MQIMPO_CONVERT_VALUE angegeben ist und die Konvertierung erfolgreich war, hat das Feld *ReturnedEncoding* bei der Rückgabe denselben Wert wie der eingegebene Wert.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

ReturnedName (MQCHARV)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld ReturnedName

Der tatsächliche Name der abgefragten Eigenschaft.

Bei der Eingabe kann über das Feld *VSPtr* oder *VSOffset* der MQCHARV-Struktur ein String Buffer übergeben werden. Die Länge des String Buffers wird über das Feld *VSBufsize* in der MQCHARV-Struktur angegeben.

Bei Rückgabe des MQINQMP-Aufrufs wird der Name der abgefragten Eigenschaft in den Zeichenfolgepuffer eingefügt, sofern der Puffer groß genug ist, um den Namen ganz aufzunehmen. In das Feld *VSLength* der MQCHARV-Struktur wird die Länge des Eigenschaftsnamens eingegeben. In das Feld *VSCCSID* der MQCHARV-Struktur wird der Zeichensatz des zurückgegebenen Namens eingegeben und es wird angezeigt, ob die Konvertierung des Namens fehlgeschlagen ist oder nicht.

Dies ist ein Ein-/Ausgabefeld. Der Anfangswert dieses Felds ist MQCHARV_DEFAULT.

StrucId (MQCHAR4)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld StrucId

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQIMPO_STRUC_ID
ID der Struktur von Optionen zum Abfragen von Nachrichteneigenschaften.

Für die Programmiersprache C wird auch die Konstante MQIMPO_STRUC_ID_ARRAY definiert; diese hat denselben Wert wie MQIMPO_STRUC_ID, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQIMPO_STRUC_ID.

TypeString (MQCHAR8)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld TypeString

Eine Zeichenfolgedarstellung des Datentyps der Eigenschaft.

Wurde die Eigenschaft in einem MQRFH2-Header angegeben und wird das MQRFH2-Attribut `dt` nicht erkannt, kann über dieses Feld der Datentyp der Eigenschaft bestimmt werden. *TypeString* wird im codierten Zeichensatz 1208 (UTF-8) zurückgemeldet und besteht aus den ersten 8 Byte des Attributwerts von `dt` der Eigenschaft, die nicht erkannt werden konnte.

Dies ist stets ein Ausgabefeld. Der Anfangswert dieses Felds besteht aus der Nullzeichenfolge in der Programmiersprache C und aus 8 Leerzeichen in anderen Programmiersprachen.

Version (MQLONG)
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Feld Version

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQIMPO_VERSION_1
Versionsnummer der Struktur von Optionen zum Abfragen von Nachrichteneigenschaften.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQIMPO_CURRENT_VERSION
Aktuelle Version der Optionsstruktur zur Abfrage von Nachrichteneigenschaften.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQIMPO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQIMPO
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Anfangswerte

Tabelle 512. Anfangswerte der Felder in MQIPMO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQIMPO_STRUC_ID	'IMPO'
Version	MQIMPO_VERSION_1	1
Options	MQIMPO_INQ_FIRST	
RequestedEncoding	MQENC_NATIVE	
RequestedCCSID	MQCCSI_APPL	
ReturnedEncoding	MQENC_NATIVE	
ReturnedCCSID	0	
Reserved1	0	
ReturnedName	MQCHARV_DEFAULT	
TypeString	Nullzeichenfolge oder Leerzeichen.	

Anmerkungen:

1. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
2. In der Programmiersprache C enthält die MakrovariableMQIMPO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

Deklaration in Programmiersprache C
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Deklaration für Programmiersprache C

```
typedef struct tagMQIMPO MQIMPO;
struct tagMQIMPO {
  MQCHAR4  StrucId;            /* Structure identifier */
  MQLONG   Version;            /* Structure version number */
  MQLONG   Options;            /* Options that control the action of
                                  MQINQMP */
  MQLONG   RequestedEncoding;  /* Requested encoding of Value */
  MQLONG   RequestedCCSID;     /* Requested character set identifier
                                  of Value */
  MQLONG   ReturnedEncoding;   /* Returned encoding of Value */
  MQLONG   ReturnedCCSID;      /* Returned character set identifier
```

```
                                   of Value */
  MQCHAR   Reserved1          /* Reserved field */
  MQCHARV  ReturnedName;      /* Returned property name */
  MQCHAR8  TypeString;        /* Property data type as a string */
};
```

COBOL-Delaration
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Deklaration für Programmiersprache COBOL

```
**   MQIMPO structure
  10 MQIMPO.
**    Structure identifier
   15 MQIMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQIMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQINQMP
   15 MQIMPO-OPTIONS               PIC S9(9) BINARY.
**    Requested encoding of VALUE
   15 MQIMPO-REQUESTEDENCODING     PIC S9(9) BINARY.
**    Requested character set identifier of VALUE
   15 MQIMPO-REQUESTEDCCSID        PIC S9(9) BINARY.
**    Returned encoding of VALUE
   15 MQIMPO-RETURNEDENCODING      PIC S9(9) BINARY.
**    Returned character set identifier of VALUE
   15 MQIMPO-RETURNEDCCSID         PIC S9(9) BINARY.
**    Reserved field
   15 MQIMPO-RESERVED1
**    Returned property name
   15 MQIMPO-RETURNEDNAME.
**     Address of variable length string
    20 MQIMPO-RETURNEDNAME-VSPTR    POINTER.
**     Offset of variable length string
    20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
**     CCSID of variable length string
    20 MQIMPO-RETURNEDNAME-VSCCSID  PIC S9(9) BINARY.
**    Property data type as string
   15 MQIMPO-TYPESTRING            PIC S9(9) BINARY.
```

Deklaration in PL/I
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Deklaration für Programmiersprache PL/I

```
dcl
 1 MQIMPO based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 Options            fixed bin(31), /* Options that control the
                                         action of MQINQMP */
  3 RequestedEncoding  fixed bin(31), /* Requested encoding of
                                         Value */
  3 RequestedCCSID     fixed bin(31), /* Requested character set
                                         identifier of Value */
  3 ReturnedEncoding   fixed bin(31), /* Returned encoding of
                                         Value */
  3 ReturnedCCSID      fixed bin(31), /* Returned character set
                                         identifier of Value */
  3 Reserved1          fixed bin(31), /* Reserved field */
  3 ReturnedName,                     /* Returned property name */
   5 ReturnedName_VSPtr      pointer,       /* Address of returned
                                               name */
   5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                               name */
    5 5 ReturnedName_VSCCSID  fixed bin(31), /* CCSID of returned
                                               name */
  3 TypeString         char(8);       /* Property data type as
                                         string */
```

Deklaration in High Level Assembler
Optionsstruktur zur Abfrage der Nachrichteneigenschaften - Deklaration für Programmiersprache Assembler

```
MQIMPO                      DSECT
MQIMPO_STRUCID              DS   CL4  Structure identifier
MQIMPO_VERSION              DS   F    Structure version number
MQIMPO_OPTIONS              DS   F    Options that control the
*                                     action of MQINQMP
```

```
MQIMPO_REQUESTEDENCODING     DS   F    Requested encoding of VALUE
MQIMPO_REQUESTEDCCSID        DS   F    Requested character set
*                                      identifier of VALUE
MQIMPO_RETURNEDENCODING      DS   F    Returned encoding of VALUE
MQIMPO_RETURNEDCCSID         DS   F    Returned character set
*                                      identifier of VALUE
MQIMPO_RESERVED1             DS   F    Reserved field
MQIMPO_RETURNEDNAME          DS   0F   Force fullword alignment
MQIMPO_RETURNEDNAME_VSPTR    DS   F    Address of returned name
MQIMPO_RETURNEDNAME_VSOFFSET DS   F    Offset of returned name
MQIMPO_RETURNEDNAME_VSLENGTH DS   F    Length of returned name
MQIMPO_RETURNEDNAME_VSCCSID  DS   F    CCSID of returned name
MQIMPO_RETURNEDNAME_LENGTH   EQU  *-MQIMPO_RETURNEDNAME
                             ORG  MQIMPO_RETURNEDNAME
MQIMPO_RETURNEDNAME_AREA     DS   CL(MQIMPO_RETURNEDNAME_LENGTH)
*
MQIMPO_TYPESTRING            DS   CL8  Property data type as string
MQIMPO_LENGTH                EQU  *-MQIMPO
MQIMPO_AREA                  DS   CL(MQIMPO_LENGTH)
```

MQMD - Nachrichtendeskriptor

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 513. Felder im MQMD

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Report	Optionen für Berichtsnachrichten	Bericht
MsgType	Nachrichtentyp	MsgType
Expiry	Nachrichtenlebensdauer	MQMD - Expiry-Feld
Feedback	Rückmeldung oder Ursachencode	MQMD - Feedback-Feld
Encoding	Numerische Codierung von Nachrichtendaten	Encoding
CodedCharSetId	Zeichensatzkennung von Nachrichtendaten	CodedCharSetId
Format	Name des Formats von Nachrichtendaten	Format
Priority	Nachrichtenpriorität	Priorität
Persistence	Nachrichtenpersistenz	Permanenz
MsgId	Nachrichten-ID	MQMD - MsgId-Feld
CorrelId	Korrelations-ID	CorrelId
BackoutCount	Zurücksetzungszähler	BackoutCount
ReplyToQ	Name der Antwortwarteschlange	ReplyToQ
ReplyToQMgr	Name des Antwortwarteschlangenmanagers	ReplyToQMgr
UserIdentifier	Benutzer-ID	UserIdentifier
AccountingToken	Abrechnung	AccountingToken
ApplIdentityData	Anwendungsdaten zur Identität	ApplIdentityData
PutApplType	Typ der Anwendung, die die Nachricht eingereiht hat	PutApplType
PutApplName	Name der Anwendung, die die Nachricht eingereiht hat	PutApplName

Tabelle 513. Felder im MQMD (Forts.)		
Feld	**Beschreibung**	**Thema**
PutDate	Datum der Nachrichteneinreihung	PutDate
PutTime	Uhrzeit, zu der die Nachricht eingereiht wurde	PutTime
ApplOriginData	Anwendungsdaten zum Ursprung	ApplOriginData
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQMD_VERSION_2 ist.		
GroupId	Gruppen-ID	GroupId
MsgSeqNumber	Folgenummer einer logischen Nachricht innerhalb einer Gruppe	MsgSeqNumber
Offset	Relative Adresse von Daten in einer physischen Nachricht ab dem Anfang der logischen Nachricht	Offset
MsgFlags	Nachrichtenmarkierungen	MQMD - MsgFlags-Feld
OriginalLength	Länge der ursprünglichen Nachricht	OriginalLength

Überblick für MQMD

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQMD-Struktur enthält die Steuerinformationen, die zusammen mit den Anwendungsdaten versendet werden, wenn eine Nachricht zwischen sendenden und empfangenden Anwendungen unterwegs ist. Die Struktur ist ein Ein-/Ausgabe-Parameter bei den MQGET-, MQPUT- und MQPUT1-Aufrufen.

Version: Die aktuelle Version von MQMD ist MQMD_VERSION_2. Anwendungen, die auf mehrere Umgebungen portierbar sein sollen, müssen sicherstellen, dass die erforderliche Version von MQMD von allen betroffenen Umgebungen unterstützt wird. Felder, die nur in den neueren Versionen der Struktur vorhanden sind, werden in den nachfolgenden Beschreibungen entsprechend gekennzeichnet.

Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützten Programmiersprachen bereitgestellt werden, enthalten die aktuellste Version von MQMD, die von der Umgebung unterstützt wird, wobei der Anfangswert des Feldes *Version* auf MQMD_VERSION_1 gesetzt ist. Um Felder zu verwenden, die in der Struktur der Version 1 nicht verfügbar sind, muss die Anwendung das Feld *Version* auf die Versionsnummer der erforderlichen Version setzen.

Eine Deklaration für die Struktur Version-1 steht unter dem Namen MQMD1 zur Verfügung.

Zeichensatz und Codierung: Daten in dem MQMD müssen dem Zeichensatz und der Codierung des lokalen Warteschlangenmanagers entsprechen; diese werden von dem Warteschlangenmanagerattribut *CodedCharSetId* und von MQENC_NATIVE vorgegeben. Wenn die Anwendung jedoch als WebSphere MQ-MQI-Client ausgeführt wird, muss die Struktur dem Zeichensatz und der Codierung des Clients entsprechen.

Wenn sendender und empfangender Warteschlangenmanager unterschiedliche Zeichensätze oder Codierungen verwenden, werden die Daten in MQMD automatisch konvertiert. Es ist nicht notwendig, dass die Anwendung den MQMD konvertiert.

Verwenden verschiedener Versionen von MQMD: Ein MQMD Version-2 entspricht der Verwendung eines MQMD Version-1 zusammen mit einer Nachricht, der eine MQMDE-Struktur vorangestellt ist. Wenn jedoch alle Felder in der MQMDE-Struktur ihren Standardwerten entsprechen, kann die MQMDE ausgeschlossen werden. Ein MQMD Version-1 mit einer MQMDE wird wie folgt verwendet:

- Wenn die Anwendung in MQPUT- und MQPUT1-Aufrufen einen MQMD Version-1 bereitstellt, besteht für die Anwendung die Möglichkeit, den Nachrichtendaten eine MQMDE voranzustellen, indem das *Format*-Feld im MQMD auf MQFMT_MD_EXTENSION gesetzt wird, um anzuzeigen, dass eine MQMDE

vorhanden ist. Wenn die Anwendung keine MQMDE angibt, setzt der Warteschlangenmanager Standardwerte für die Felder in der MQMDE voraus.

Anmerkung: Einige der Felder, die im MQMD Version-2 vorliegen, aber nicht im MQMD Version-1, sind bei MQPUT- und MQPUT1-Aufrufen Ein-/Ausgabefelder. Der Warteschlangenmanager gibt jedoch als Ausgabe der MQPUT- und MQPUT1- Aufrufe *keine* Werte in den entsprechenden Feldern in der MQMDE zurück. Wenn für die Anwendung diese Ausgabewerte erforderlich sind, muss ein MQMD Version-2 vorliegen.

- Wenn die Anwendung beim MQGET-Aufruf einen MQMD der Version 1 angibt, stellt der Warteschlangenmanager der zurückgegebenen Nachricht eine MQMDE voran. Dazu muss jedoch mindestens eines der Felder in der MQMDE einen Wert aufweisen, der kein Standardwert ist. Das *Format*-Feld im MQMD hat den Wert MQFMT_MD_EXTENSION, um anzuzeigen, dass eine MQMDE vorhanden ist.

Die vom Warteschlangenmanager für die Felder in der MQMDE verwendeten Standardwerte entsprechen den Anfangswerten dieser Felder, die in Tabelle 518 auf Seite 458 angezeigt werden.

Wenn eine Nachricht in eine Übertragungswarteschlange eingereiht ist, werden einige Felder im MQMD auf bestimmte Werte gesetzt; weitere Informationen finden Sie im Abschnitt „MQXQH – Header der Übertragungswarteschlange" auf Seite 609.

Nachrichtenkontext: Bestimmte Felder im MQMD enthalten den Nachrichtenkontext. Es gibt zwei Arten von Nachrichtenkontext: *Identitätskontext* und *Ursprungskontext*. Üblicherweise:

- Bezieht sich Identitätskontext auf die Anwendung, die *ursprünglich* die Nachricht eingereiht hat
- Bezieht sich Ursprungskontext auf die Anwendung, die die Nachricht *zuletzt* eingereiht hat.

Bei diesen beiden Anwendungen kann es sich um dieselbe Anwendung, aber auch um unterschiedliche Anwendungen handeln (z. B. wenn eine Nachricht von einer Anwendung an eine andere weitergeleitet wird).

Obwohl Identitätskontext und Ursprungskontext typischerweise die beschriebenen Bedeutungen haben, hängt der Inhalt beider Arten von Kontextfeldern im MQMD von den MQPMO_*_CONTEXT-Optionen ab, die angegeben werden, wenn die Nachricht eingereiht wird. Aus diesem Grund steht der Identitätskontext nicht zwingend mit der Anwendung, die ursprünglich die Nachricht eingereiht hat, in Beziehung und der Ursprungskontext steht nicht unbedingt in Beziehung zu der Anwendung, die die Nachricht zuletzt eingereiht hat. Es hängt von dem Design der Anwendungssuite ab.

Der Nachrichtenkanalagent (MCA) verändert nie den Nachrichtenkontext. Nachrichtenkanalagenten, die Nachrichten von fernen Warteschlangenmanagern erhalten, verwenden die Kontextoption MQPMO_SET_ALL_CONTEXT beim MQPUT- und beim MQPUT1-Aufruf. Dies ermöglicht dem empfangenden Nachrichtenkanalagenten, genau den Nachrichtenkontext zu erhalten, der zusammen mit der Nachricht von dem sendenden Nachrichtenkanalagenten gesendet wurde. Dies führt jedoch dazu, dass der Ursprungskontext weder zu dem Nachrichtenkanalagenten, der die Nachricht gesendet hat, noch zu dem Nachrichtenkanalagenten, der die Nachricht empfangen hat, in Beziehung steht. Der Ursprungskontext bezieht sich auf eine frühere Anwendung, die die Nachricht einreihte. Wenn alle temporären Anwendungen den Nachrichtenkontext übergeben haben, bezieht sich der Ursprungskontext auf die ursprüngliche Anwendung.

In den Beschreibungen werden die Kontextfelder so beschrieben, als würden sie so eingesetzt, wie bereits beschrieben. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Felder für MQMD

Die MQMD-Struktur enthält die folgenden Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

Tabelle 514. Felder im MQMD

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId

Tabelle 514. Felder im MQMD (Forts.)		
Feld	**Beschreibung**	**Thema**
Version	Strukturversionsnummer	Version
Report	Optionen für Berichtsnachrichten	Bericht
MsgType	Nachrichtentyp	MsgType
Expiry	Nachrichtenlebensdauer	MQMD - Expiry-Feld
Feedback	Rückmeldung oder Ursachencode	MQMD - Feedback-Feld
Encoding	Numerische Codierung von Nachrichtendaten	Encoding
CodedCharSetId	Zeichensatzkennung von Nachrichtendaten	CodedCharSetId
Format	Name des Formats von Nachrichtendaten	Format
Priority	Nachrichtenpriorität	Priorität
Persistence	Nachrichtenpersistenz	Permanenz
MsgId	Nachrichten-ID	MQMD - MsgId-Feld
CorrelId	Korrelations-ID	CorrelId
BackoutCount	Zurücksetzungszähler	BackoutCount
ReplyToQ	Name der Antwortwarteschlange	ReplyToQ
ReplyToQMgr	Name des Antwortwarteschlangenmanagers	ReplyToQMgr
UserIdentifier	Benutzer-ID	UserIdentifier
AccountingToken	Abrechnung	AccountingToken
ApplIdentityData	Anwendungsdaten zur Identität	ApplIdentityData
PutApplType	Typ der Anwendung, die die Nachricht eingereiht hat	PutApplType
PutApplName	Name der Anwendung, die die Nachricht eingereiht hat	PutApplName
PutDate	Datum der Nachrichteneinreihung	PutDate
PutTime	Uhrzeit, zu der die Nachricht eingereiht wurde	PutTime
ApplOriginData	Anwendungsdaten zum Ursprung	ApplOriginData
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQMD_VERSION_2 ist.		
GroupId	Gruppen-ID	GroupId
MsgSeqNumber	Folgenummer einer logischen Nachricht innerhalb einer Gruppe	MsgSeqNumber
Offset	Relative Adresse von Daten in einer physischen Nachricht ab dem Anfang der logischen Nachricht	Offset
MsgFlags	Nachrichtenmarkierungen	MQMD - MsgFlags-Feld
OriginalLength	Länge der ursprünglichen Nachricht	OriginalLength

AccountingToken (MQBYTE32)

Dies ist das Abrechnungstoken und Teil des **Identitätskontexts** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten „Überblick für MQMD” auf Seite 400 und Nachrichtenkontext.

AccountingToken ermöglicht einer Anwendung, Arbeit, die aufgrund einer Nachricht angefallen ist, in angemessener Weise in Rechnung zu stellen. Der Warteschlangenmanager behandelt diese Informationen als Bitzeichenfolge, ohne jedoch ihren Inhalt zu prüfen.

Der Warteschlangenmanager erstellt diese Information wie nachfolgend aufgeführt:

- Das erste Byte des Felds gibt die Länge der Abrechnungsinformationen in den folgenden Byte an. Die Länge hat einen Wert im Bereich von 0 bis 30 und wird als binäre Ganzzahl gespeichert.
- Das zweite Byte und die nachfolgenden Byte (entsprechend der Festlegung durch das Längenfeld) werden auf die Abrechnungsdaten gesetzt, die der Umgebung entsprechen.
  - Bei z/OS werden die Abrechnungsdaten folgendermaßen angegeben:
    - Bei z/OS-Stapelanwendungen die Abrechnungsdaten der JES-Jobkarte oder einer JES-ACCT-Anweisung in der EXEC-Karte (Kommatrennzeichen werden in X'FF' umgewandelt). Diese Informationen werden bei Bedarf auf 31 Byte gekürzt.
    - Für TSO auf die Kontonummer des Benutzers.
    - Bei CICS die Arbeitseinheiten-ID LU 6.2 (UEPUOWDS) (26 Byte).
    - Bei IMS der 8 Zeichen lange PSB-Name verkettet mit dem 16 Zeichen langen IMS-Recovery-Token.
  - Bei IBM i werden die Abrechnungsdaten auf den Berechnungscode des Jobs gesetzt.
  - Bei UNIX-Systemen werden die Abrechnungsdaten auf die numerische Benutzer-ID in ASCII-Zeichen gesetzt.
  - Bei Windows werden die Abrechnungsdaten in einem komprimierten Format auf eine Windows-Sicherheits-ID (SID) gesetzt. Die SID gibt die im *UserIdentifier*-Feld gespeicherte Benutzer-ID eindeutig an. Wenn die SID im *AccountingToken*-Feld gespeichert wird, wird die 6 Byte lange ID-Berechtigung, die sich im dritten und den darauf folgenden Bytes befindet, übergangen. Wenn beispielsweise die Windows-SID 28 Bytes lang ist, werden 22 Bytes der SID-Daten im *AccountingToken*-Feld gespeichert.
- Das letzte Byte, Byte 32, des Abrechnungsfeldes wird auf den Typ des Abrechnungstokens gesetzt (in diesem Fall MQACTT_NT_SECURITY_ID, x '0b'):

  MQACTT_CICS_LUOW_ID
  CICS-LUOW-ID.

  MQACTT_NT_SECURITY_ID
  Windows-Sicherheits-ID.

  MQACTT_OS400_ACCOUNT_TOKEN
  IBM i-Abrechnungstoken.

  MQACTT_UNIX_NUMERIC_ID
  UNIX-Systeme: numerische Kennung.

  MQACTT_USER
  Benutzerdefiniertes Abrechnungstoken.

  MQACTT_UNKNOWN
  Unbekannter Abrechnungstokentyp.

  Der Abrechnungstokentyp wird nur in den folgenden Umgebungen auf einen expliziten Wert gesetzt: AIX, HP-UX, IBM i, Solaris, Linux, Windowsplus WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind. In anderen Umgebungen wird der Abrechnungstokentyp auf den Wert MQACTT_UNKNOWN gesetzt. Verwenden Sie in diesen Umgebungen das *PutApplType*-Feld, um den empfangenen Abrechnungstokentyp abzuleiten.
- Alle anderen Bytes werden auf den Wert binäre Null gesetzt.

Bei MQPUT- und MQPUT1-Aufrufen ist dies ein Ein-/Ausgabefeld, falls MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT im Parameter *PutMsgOpts* angegeben werden. Wenn weder

MQPMO_SET_IDENTITY_CONTEXT noch MQPMO_SET_ALL_CONTEXT angegeben sind, wird dieses Feld bei der Eingabe ignoriert oder ist ein Nur-Ausgabe-Feld. Weitere Informationen zum Nachrichtenkontext finden Sie unter Nachrichtenkontext .

Nach erfolgreichem Beenden des MQPUT- oder MQPUT1-Aufrufs enthält dieses Feld den Wert *AccountingToken*, der mit der Nachricht übertragen wurde, wenn sie in eine Warteschlange eingereiht wurde. Dabei handelt es sich um den Wert von *AccountingToken*, der mit der Nachricht, falls sie beibehalten wird, aufbewahrt wird (weitere Informationen zu ständigen Veröffentlichungen finden Sie in der Beschreibung von MQPMO_RETAIN in „MQPMO-Optionen (MQLONG)" auf Seite 491), aber nicht als *AccountingToken* verwendet wird, wenn die Nachricht als Veröffentlichung an Subskribenten gesendet wird, denn sie stellen einen Wert bereit, *AccountingToken* in allen, an sie gesandten Veröffentlichungen außer Kraft zu setzen. Wenn die Nachricht keinen Kontext aufweist, enthält das Feld nur eine binäre Null.

Dies ist ein Ausgabefeld für den MQGET-Aufruf.

Dieses Feld wird in keiner Weise basierend auf dem Zeichensatz des Warteschlangenmanagers konvertiert; das Feld wird wie eine Bitzeichenfolge und nicht wie eine Zeichenfolge behandelt.

Der Warteschlangenmanager verwendet die Informationen dieses Felds nicht. Die Anwendung muss die Informationen interpretieren, falls sie sie für Abrechnungszwecke verwenden will.

Sie können für das *AccountingToken*-Feld die folgenden Sonderwerte verwenden:

MQACT_NONE
Es ist kein Abrechnungstoken angegeben.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQACT_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQACT_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_ACCOUNTING_TOKEN_LENGTH angegeben. Der Anfangswert dieses Felds ist MQACT_NONE.

ApplIdentityData (MQCHAR32)

Dieses Attribut ist Bestandteil des **Identitätskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie unter „Überblick für MQMD" auf Seite 400 und Nachrichtenkontext .

ApplIdentityData enthält Informationen, die in der Anwendungssuite definiert wurden, und kann zusätzliche Informationen zur Nachricht oder zum Absender zur Verfügung stellen. Der Warteschlangenmanager behandelt diese Informationen als Zeichendaten, ohne jedoch ihr Format zu definieren. Wenn der Warteschlangenmanager diese Informationen erstellt, sind sie gänzlich leer.

Bei MQPUT- und MQPUT1-Aufrufen ist dies ein Ein-/Ausgabefeld, falls MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT im Parameter *PutMsgOpts* angegeben werden. Wenn ein Nullzeichen vorliegt, werden das Nullzeichen und jegliche nachfolgenden Zeichen vom Warteschlangenmanager in Leerzeichen umgewandelt. Wenn weder MQPMO_SET_IDENTITY_CONTEXT noch MQPMO_SET_ALL_CONTEXT angegeben sind, wird dieses Feld bei der Eingabe ignoriert oder ist ein Nur-Ausgabe-Feld. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Nach erfolgreichem Beenden des MQPUT- oder MQPUT1-Aufrufs enthält dieses Feld den Wert *ApplIdentityData*, der mit der Nachricht übertragen wurde, wenn sie in eine Warteschlange eingereiht wurde. Dabei handelt es sich um den Wert von *ApplIdentityData*, der mit der Nachricht, falls sie beibehalten wird, aufbewahrt wird (weitere Informationen zu ständigen Veröffentlichungen finden Sie in der Beschreibung von MQPMO_RETAIN), aber nicht als *ApplIdentityData* verwendet wird, wenn die Nachricht als Veröffentlichung an Subskribenten gesendet wird, denn sie stellen einen Wert bereit, *ApplIdentityData* in allen, an sie gesandten Veröffentlichungen außer Kraft zu setzen. Wenn die Nachricht keinen Kontext aufweist, ist das Feld leer.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Die Länge dieses Felds wird durch MQ_APPL_IDENTITY_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 32 Leerzeichen in anderen Programmiersprachen.

ApplOriginData (MQCHAR4)

Dieses Attribut ist Bestandteil des **Ursprungskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie unter „Überblick für MQMD" auf Seite 400 und Nachrichtenkontext .

ApplOriginData wird von der Anwendungssuite vorgegeben; über dieses Attribut können zusätzliche Informationen zum Ursprung der Nachricht bereitgestellt werden. Es kann beispielsweise von Anwendungen eingestellt werden, die mit geeigneten Benutzerberechtigungen ausgeführt werden, um anzuzeigen, ob die Identitätsdaten vertrauenswürdig sind.

Der Warteschlangenmanager behandelt diese Informationen als Zeichendaten, ohne jedoch ihr Format zu definieren. Wenn der Warteschlangenmanager diese Informationen erstellt, sind sie gänzlich leer.

Bei den MQPUT- und MQPUT1-Aufrufen ist es, falls MQPMO_SET_ALL_CONTEXT im *PutMsgOpts*-Parameter angegeben wird, ein Ein-/Ausgabefeld. Alle Informationen, die in dem Feld auf ein Nullzeichen folgen, werden verworfen. Der Warteschlangenmanager wandelt das Nullzeichen und alle anschließenden Zeichen in Leerzeichen um. Wenn MQPMO_SET_ALL_CONTEXT nicht angegeben wird, wird dieses Feld bei der Eingabe ignoriert und ist ein Nur-Ausgabe-Feld.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Die Länge dieses Felds wird durch MQ_APPL_ORIGIN_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist in der Programmiersprache C die Nullzeichenfolge und in anderen Programmiersprachen sind es 4 Leerzeichen.

Wenn die Nachricht veröffentlicht wird, obwohl *ApplOriginData* festgelegt ist, bleibt sie in der Subskription, die sie empfängt, leer.

BackoutCount (MQLONG)

Dieses Feld gibt an, wie häufig die Nachricht bisher vom MQGET-Aufruf als Teil einer Arbeitseinheit zurückgegeben und anschließend zurückgesetzt wurde. Es hilft der Anwendung dabei, Verarbeitungsfehler festzustellen, die auf Nachrichteninhalten basieren. Bei der Zählung werden MQGET-Aufrufe, die eine der MQGMO_BROWSE_*-Optionen angeben, nicht berücksichtigt.

Auf die Genauigkeit dieses Zählers wirkt sich das Warteschlangenattribut *HardenGetBackout* aus (siehe „Attribute für Warteschlangen" auf Seite 833).

Bei z/OS bedeutet der Wert 255, dass die Nachricht mindestens 255-mal zurückgesetzt wurde; der zurückgegebene Wert ist niemals größer als 255.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Bei MQPUT- und MQPUT1-Aufrufen wird es ignoriert. Der Anfangswert dieses Felds ist `0`.

CodedCharSetId (MQLONG)
Dieses Feld gibt die Zeichensatzkennung der Zeichendaten des Nachrichtentextes an.

Anmerkung: Zeichendaten im MQMD sowie die weiteren MQ-Datenstrukturen, die Parameter bei Aufrufen darstellen, müssen dem Zeichensatz des Warteschlangenmanagers entsprechen. Dieser Wert wird durch das Attribut *CodedCharSetId* des Warteschlangenmanagers definiert; weitere Informationen zu diesem Attribut finden Sie im Abschnitt „Attribute für den Warteschlangenmanager" auf Seite 797 .

Wenn dieses Feld beim Aufrufen von MQGET mit MQGMO_CONVERT in den Optionen auf MQCCSI_Q_MGR gesetzt ist, unterscheidet sich das Verhalten der Client- und Serveranwendungen. Bei Serveranwendungen ist die Codepage, die für die Zeichenkonvertierung verwendet wird, die *CodedCharSetId* des Warteschlangenmanagers; bei Clientanwendungen ist die Codepage, die für die Zeichenkonvertierung verwendet wird, die Codepage der aktuellen Ländereinstellung.

Für Clientanwendungen wird MQCCSI_Q_MGR eingegeben, auf Basis der Ländereinstellung des Clients statt der des Warteschlangenmanagers. Die Ausnahme von dieser Regel ist, wenn Sie eine Nachricht in eine IMS -Brückenwarteschlange einreihen; was im MQMD-Feld *CodedCharSetId* zurückgegeben wird, ist die CCSID des Warteschlangenmanagers.

Der folgende Sonderwert darf nicht verwendet werden:

MQCCSI_APPL
Dies führt zu einem falschen Wert im MQMD-Feld *CodedCharSetId* und verursacht den Rückkehrcode MQRC_SOURCE_CCSID_ERROR (oder MQRC_FORMAT_ERROR für z/OS), wenn die Nachricht mit dem MQGET-Aufruf mit der MQGMO_CONVERT-Option empfangen wird.

Sie können die folgenden Sonderwerte verwenden:

MQCCSI_Q_MGR
Die Zeichendaten der Nachricht entsprechen dem Zeichensatz des Warteschlangenmanagers.

Bei MQPUT- und MQPUT1-Aufrufen ändert der Warteschlangenmanager diesen Wert in dem MQMD, der zusammen mit der Nachricht verwendet wird, zur wahren Zeichensatzkennung des Warteschlangenmanagers. Dies führt dazu, dass MQCCSI_Q_MGR nie vom MQGET-Aufruf zurückgegeben wird.

MQCCSI_DEFAULT
Die *CodedCharSetId* der Daten im Feld *String* wird durch das Feld *CodedCharSetId* in der Headerstruktur definiert, die der MQCFH-Struktur vorausgeht, oder durch das Feld *CodedCharSetId* im MQMD, wenn sich der MQCFH am Anfang der Nachricht befindet.

MQCCSI_INHERIT
Die Zeichendaten der Nachricht entsprechen den Zeichendaten dieser Struktur; es handelt sich um den Zeichensatz des Warteschlangenmanagers. Einzig für den MQMD hat MQCCSI_INHERIT dieselbe Bedeutung wie MQCCSI_Q_MGR.

Der Nachrichtenmanager ändert diesen Wert in dem MQMD, der zusammen mit der Nachricht gesendet wird, zur tatsächlichen Zeichensatzkennung des MQMD. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

Verwenden Sie nicht MQCCSI_INHERIT, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

MQCCSI_EMBEDDED
Die Zeichendaten der Nachricht verwenden denselben Zeichensatz wie die ID, die sich in den Nachrichtendaten selbst befindet. In den Nachrichtendaten kann eine beliebige Zahl an Zeichensatzkennungen integriert sein, die jeweils auf verschiedene Teile der Daten angewendet werden. Dieser Wert muss für PCF-Nachrichten - im Format MQFMT_ADMIN, MQFMT_EVENT oder MQFMT_PCF - verwendet werden, die Daten mit verschiedenen Zeichensätzen enthalten. Für jede MQCFST-, MQCFSL- und MQCFSF-Struktur in der PCF-Nachricht muss statt MQCCSI_DEFAULT eine explizite Zeichensatzkennung angegeben sein.

Sollte eine Nachricht mit dem Format MQFMT_EMBEDDED_PCF Daten mit verschiedenen Zeichensätzen enthalten, verwenden Sie MQCCSI_EMBEDDED nicht. Setzen Sie stattdessen das Flags-Feld in der MQEPH-Struktur auf MQEPH_CCSID_EMBEDDED. Dies entspricht der Einstellung MQCCSI_EMBEDDED in der führenden Struktur. Für jede MQCFST-, MQCFSL- und MQCFSF-Struktur in der PCF-Nachricht muss dann statt MQCCSI_DEFAULT eine explizite Zeichensatzkennung angegeben sein. Weitere Informationen zur MQEPH-Struktur finden Sie unter „MQEPH - Eingebetteter PCF-Header” auf Seite 342.

Geben Sie diesen Wert nur bei MQPUT- und MQPUT1-Aufrufen an. Wenn er beim MQGET-Aufruf angegeben wird, verhindert er die Konvertierung der Nachricht.

Beim MQPUT-Aufruf und beim MQPUT1-Aufruf ändert der Warteschlangenmanager die Werte MQCCSI_Q_MGR und MQCCSI_INHERIT im MQMD, der zusammen mit der Nachricht verwendet wurde, wie oben beschrieben; den MQMD, der beim MQPUT- oder beim MQPUT1-Aufruf angegeben wird, ändert er jedoch nicht. Der angegebene Wert wird ansonsten keiner weiteren Prüfung unterzogen.

Anwendungen, die Nachrichten erhalten, müssen dieses Feld auf den Wert überprüfen, der von der Anwendung erwartet wird; sollten sich die Werte unterscheiden, muss die Anwendung gegebenenfalls die Zeichendaten in der Nachricht ändern.

Wenn Sie die MQGMO_CONVERT-Option beim MQGET-Aufruf angeben, handelt es sich bei diesem Feld um ein Ein-/Ausgabefeld. Der Wert entspricht der ID des codierten Zeichensatzes, in den die Nachrichtendaten nötigenfalls konvertiert werden. Wenn die Konvertierung erfolgreich oder unnötig ist, bleibt der Wert - abgesehen davon, dass MQCCSI_Q_MGR oder MQCCSI_INHERIT in den tatsächlichen Wert

umgewandelt werden - unverändert. Ist die Konvertierung nicht erfolgreich, stellt der Wert nach dem MQGET-Aufruf die ID des codierten Zeichensatzes der konvertierten Nachricht dar, die an die Anwendung zurückgegeben wird.

Andernfalls ist dies ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQCCSI_Q_MGR.

CorrelId (MQBYTE24)
Das CorrelId-Feld ist eine Eigenschaft des Nachrichtenheaders, die dazu verwendet werden kann, eine bestimmte Nachricht oder Nachrichtengruppe zu ermitteln.

Dies ist eine Bytefolge, mit der die Anwendung eine Nachricht mit einer anderen Nachricht oder Arbeit verknüpfen kann, die ebenfalls von der Anwendung ausgeführt wird. Die Korrelations-ID ist eine persistente Eigenschaft der Nachricht, die auch bei einem Neustart des Warteschlangenmanagers bestehen bleibt. Da die Korrelations-ID eine Bytefolge und keine Zeichenfolge ist, wird sie *nicht* konvertiert, wenn die Nachricht von einem Warteschlangenmanager zu einem anderen weitergeleitet wird.

Bei dem MQPUT- und dem MQPUT1-Aufruf kann die Anwendung einen beliebigen Wert angeben. Der Warteschlangenmanager überträgt den Wert zusammen mit der Nachricht und übermittelt ihn der Anwendung, die die Abrufanforderung für die Nachricht absetzt.

Falls die Anwendung MQPMO_NEW_CORREL_ID angibt, erstellt der Warteschlangenmanager eine eindeutige Korrelations-ID, die mit der Nachricht gesendet und außerdem an die sendende Anwendung bei der Ausgabe mit dem MQPUT- oder MQPUT1-Aufruf zurückgegeben wird.

Eine vom Warteschlangenmanager erstellte Korrelations-ID besteht aus einer drei Byte langen Produkt-ID (AMQ oder CSQ in ASCII oder EBCDIC), gefolgt von einem reservierten Byte und einer produktspezifischen Implementierung einer eindeutigen Zeichenfolge. Bei WebSphere MQ enthält diese produktspezifische Implementierungszeichenfolge die ersten 12 Zeichen des Namens des Warteschlangenmanagers und einen Wert, der von der Systemuhr abgeleitet wurde. Damit sichergestellt wird, dass die Nachrichten-IDs eindeutig sind, müssen die Namen sämtlicher miteinander kommunizierender Warteschlangenmanager sich in den ersten 12 Zeichen unterscheiden. Die Fähigkeit, eine eindeutige Zeichenfolge zu generieren, ist außerdem davon abhängig, dass die Systemuhr nicht zurückgedreht wird. Um die Möglichkeit auszuschließen, dass eine Nachrichten-ID generiert wird, indem der Warteschlangenmanager eine Nachrichten-ID dupliziert, die von der Anwendung generiert wurde, darf die Anwendung keine IDs generieren, deren Anfangszeichen im Bereich A bis I in ASCII oder EBCDIC liegen (X'41' bis X'49' und X'C1' bis X'C9'). Allerdings wird die Anwendung nicht davon abgehalten, IDs mit Anfangszeichen in diesen Bereichen zu erstellen.

Diese erstellte Korrelations-ID wird mit der Nachricht zusammen aufbewahrt, falls sie beibehalten wird, und wenn die Nachricht als Veröffentlichung an Subskribenten gesendet wird, die MQCI_NONE im SubCorrelId-Feld des MQSD angeben, der beim MQSUB-Aufruf weitergegeben wird, wird sie als Korrelations-ID benutzt. Weitere Informationen zu ständigen Veröffentlichungen finden Sie im Abschnitt über MQPMO-Optionen.

Wenn der Warteschlangenmanager oder wenn ein Nachrichtenkanalagent eine Berichtsnachricht generiert, setzt er das *CorrelId*-Feld so wie vom *Report*-Feld der ursprünglichen Nachricht angegeben entweder auf MQRO_COPY_MSG_ID_TO_CORREL_ID oder MQRO_PASS_CORREL_ID. Anwendungen, die Berichtsnachrichten erstellen, müssen dies ebenfalls tun.

Beim MQGET-Aufruf ist *CorrelId* eines von fünf Feldern, die dazu verwendet werden können, eine bestimmte Nachricht auszuwählen, die von der Warteschlange abgerufen werden soll. Weitere Informationen dazu, wie sie Werte für dieses Feld angeben, finden Sie in der Beschreibung des *MsgId*-Felds.

Geben Sie MQCI_NONE als Korrelations-ID an, hat das dieselbe Wirkung, als würden sie MQMO_MATCH_CORREL_ID *nicht* angeben, d. h. *jede beliebige* Korrelations-ID führt zu einer Übereinstimmung.

Wenn im *GetMsgOpts*-Parameter beim MQGET-Aufruf die Option MQGMO_MSG_UNDER_CURSOR angegeben wird, wird dieses Feld ignoriert.

Bei Rückgabe des MQGET-Aufrufs wird das *CorrelId*-Feld auf die, falls vorhanden, Korrelations-ID der zurückgegebenen Nachricht gesetzt.

Die folgenden Sonderwerte können verwendet werden:

MQCI_NONE
Keine Korrelations-ID angegeben

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQCI_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQCI_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQCI_NEW_SESSION
Nachricht ist Start einer neuen Sitzung

Von der CICS-Brücke wird dieser Wert als Anfang einer neuen Sitzung angesehen, d.h. als Anfang einer neuen Folge von Nachrichten.

Für die Programmiersprache C ist auch die Konstante MQCI_NEW_SESSION_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQCI_NEW_SESSION, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Beim MQGET-Aufruf handelt es sich um ein Ein-/Ausgabefeld. Bei MQPUT- und MQPUT1-Aufrufen ist dies ein Eingabefeld, wenn MQPMO_NEW_CORREL_ID *nicht* angegeben wird und ein Ausgabefeld, wenn MQPMO_NEW_CORREL_ID *angegeben* wird. Die Länge dieses Felds wird durch MQ_CORREL_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQCI_NONE.

Anmerkung:

In einer Hierarchie können Sie die Korrelations-ID einer Veröffentlichung nicht übergeben. Das Feld wird vom Warteschlangenmanager verwendet.

Encoding (MQLONG)

Dies gibt die numerische Codierung der numerischen Daten in der Nachricht an; es wird nicht auf numerische Daten in der MQMD-Struktur selbst angewendet. Die numerische Codierung definiert die Darstellung, die für binäre Ganzzahlen, gepackte dezimale Ganzzahlen und Gleitkommazahlen verwendet wird.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Der Warteschlangenmanager prüft nicht, ob das Feld gültig ist. Der folgende spezielle Wert ist definiert:

MQENC_NATIVE
Die Codierung entspricht der Standardcodierung für die Programmiersprache und das System, auf dem die Anwendung ausgeführt wird.

Anmerkung: Der Wert dieser Konstante hängt von der Programmiersprache und der Umgebung ab. Daher müssen Anwendungen mit den für die Umgebung, in der die Anwendungen ausgeführt wird, geeigneten Header-, Makro-, COPY- und INCLUDE-Dateien kompiliert werden.

Anwendungen, die Nachrichten einreihen, geben normalerweise MQENC_NATIVE an. Anwendungen, die Nachrichten erhalten, müssen dieses Feld auf den Wert MQENC_NATIVE prüfen; wenn die Werte sich unterscheiden, muss die Anwendung eventuell numerische Daten in der Nachricht konvertieren. Fordern Sie mit der MQGMO_CONVERT-Option beim Warteschlangenmanager die Konvertierung der Nachricht als Teil der Verarbeitung des MQGET-Aufrufs an. Details zur Erstellung des Felds *Encoding* finden Sie in „Maschinencodierungen” auf Seite 901 .

Wenn Sie die MQGMO_CONVERT-Option beim MQGET-Aufruf angeben, handelt es sich bei diesem Feld um ein Ein-/Ausgabefeld. Der Wert entspricht der Codierung, in die die Nachrichtendaten nötigenfalls konvertiert werden. Wenn die Konvertierung erfolgreich oder unnötig ist, bleibt der Wert unverändert. Ist die Konvertierung nicht erfolgreich, stellt der Wert nach dem MQGET-Aufruf die Codierung der nicht konvertierten Nachricht dar, die an die Anwendung zurückgegeben wird.

Andernfalls ist dies ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQENC_NATIVE.

Expiry (MQLONG)

Dies ist ein Zeitabschnitt, der in Zehntelsekunden ausgedrückt und von der Anwendung gesetzt wird, die die Nachricht einreiht. Die Nachricht kann gelöscht werden, wenn sie nicht aus der Zielwarteschlange entfernt wird, bevor dieser Zeitraum verstrichen ist.

Der Wert wird verringert, um die Zeitspanne wiederzugeben, die die Nachricht in der Zielwarteschlange sowie - falls in eine ferne Warteschlange eingereiht wird - in jeder temporären Übertragungswarteschlange verbringt. Er kann außerdem von Nachrichtenkanalagenten verringert werden, um, falls sie von Bedeutung sein sollten, Übertragungszeiten wiederzugeben. Zudem kann eine Anwendung, die diese Nachricht an eine andere Warteschlange weiterleitet, nötigenfalls den Wert verringern, wenn sie die Nachricht über einen signifikanten Zeitraum hinweg beibehalten haben sollte. Die Ablaufzeit wird jedoch nur als Näherungswert angesehen und der Wert muss nicht verringert werden, um kleine Zeitintervalle wiederzugeben.

Wenn die Nachricht von einer Anwendung unter Verwendung des MQGET-Aufrufs abgerufen wird, gibt das *Expiry*-Feld wieder, wie viel von der ursprünglichen Ablaufzeit noch verbleibt.

Nachdem die Ablaufzeit einer Nachricht verstrichen ist, kann sie vom Warteschlangemanager gelöscht werden. Bei Auftreten eines (angezeigten oder nicht angezeigten) MQGET-Aufrufs, der die Nachricht zurückgäbe, wäre sie nicht bereits abgelaufen, wird die Nachricht gelöscht. Zum Beispiel löscht ein nicht angezeigter MQGET-Aufruf mit dem auf MQMO_NONE gesetzten *MatchOptions*-Feld in MQGMO, der eine nach dem "First In/First Out"-Prinzip sortierte Warteschlange liest, alle abgelaufenen Nachrichten bis hin zur ersten nicht abgelaufenen Nachricht. Bei einer nach Priorität sortierten Warteschlange löscht derselbe Aufruf abgelaufene Nachrichten mit einer höheren Priorität und Nachrichten mit derselben Priorität, die vor der ersten nicht abgelaufenen Nachricht in der Warteschlange eingereiht wurden.

Eine Nachricht, die abgelaufen ist, wird von einem angezeigten oder nicht angezeigten MQGET-Aufruf nie an eine Anwendung zurückgegeben, so dass der Wert im *Expiry*-Feld des Nachrichtendeskriptors nach einem erfolgreichen MQGET-Aufruf entweder größer als null ist oder dem Sonderwert MQEI_UNLIMITED entspricht.

Wenn eine Nachricht in eine ferne Warteschlange eingereiht wird, läuft sie eventuell ab, während sie sich in einer temporären Übertragungswarteschlange befindet, und wird gelöscht, bevor sie die Zielwarteschlange erreicht.

Wenn eine abgelaufene Nachricht gelöscht wird, wird ein Bericht generiert, falls die Nachricht eine der MQRO_EXPIRATION_*-Berichtsoptionen angegeben hatte. Wenn keine dieser Optionen angegeben ist, wird kein Bericht erstellt; es wird davon ausgegangen, dass die Nachricht nach diesem Zeitraum nicht mehr länger relevant ist (eventuell weil sie durch eine spätere Nachricht ersetzt wurde).

Bei einer Nachricht, die mit einem Synchronisationspunkt eingereiht wurde, startet das Ablaufintervall mit dem Einreihen der Nachricht und nicht beim Festschreiben des Synchronisationspunkts. Es ist möglich, dass das Ablaufintervall abläuft, bevor der Synchronisationspunkt festgeschrieben wurde. In diesem Fall wird die Nachricht zu einem Zeitpunkt nach der Festschreibungsoperation gelöscht und nicht als Antwort auf eine MQGET-Operation an eine Anwendung zurückgegeben.

Jedes andere Programm, dass Nachrichten auf Grundlage der Ablaufzeit löscht, muss ebenfalls bei entsprechender Anforderung eine Berichtsnachricht senden.

Anmerkung:

1. Wenn eine Nachricht mit einer *Expiry*-Zeit von null oder mit einem Zeitraum größer als 999 999 999 eingereiht wurde, schlägt der MQPUT- oder MQPUT1-Aufruf mit dem Ursachencode MQRC_EXPIRY_ERROR fehl; in diesem Fall wird keine Berichtsnachricht generiert.
2. Da eine Nachricht mit einer bereits verstrichenen Ablaufzeit eventuell erst später gelöscht wird, kann es sein, dass in einer Warteschlange Nachrichten vorhanden sind, deren Ablaufzeit bereits verstrichen ist, und die aus diesem Grund nicht abgerufen werden können. Diese Nachrichten werden trotzdem für die Warteschlange immer mitgezählt und zwar zu allen Zwecken, einschließlich des Auslösertyps DEPTH.
3. Bei Anforderung wird ein Ablaufbericht erstellt, wenn die Nachricht gelöscht wird, jedoch nicht zu dem Zeitpunkt, ab dem eine Löschung zulässig ist.

4. Das Löschen einer abgelaufenen Nachricht und das Erstellen eines Ablaufberichts sind niemals Teil der Arbeitseinheit der Anwendung - auch dann nicht, wenn die Nachricht aufgrund eines innerhalb einer Arbeitseinheit arbeitenden MQGET-Aufrufs für das Löschen terminiert wurde.
5. Wenn eine Nachricht, die schon fast abgelaufen ist, von einem MQGET-Aufruf innerhalb einer Arbeitseinheit abgerufen wird und die Arbeitseinheit danach zurückgesetzt wird, kann es eventuell vorkommen, dass die Nachricht zum Löschen freigegeben wird, bevor sie erneut abgerufen werden kann.
6. Wenn eine Nachricht, die schon fast abgelaufen ist, durch einen MQGET-Aufruf mit MQGMO_LOCK gesperrt wird, kann es vorkommen, dass die Nachricht zum Löschen freigegeben wird, bevor sie von einem MQGET-Aufruf mit MQGMO_MSG_UNDER_CURSOR abgerufen werden kann; falls dies geschieht, wird der Ursachencode MQRC_NO_MSG_UNDER_CURSOR bei diesem darauf folgenden MQGET-Aufruf zurückgegeben.
7. Wird eine Anforderungsnachricht mit einer Ablaufzeit abgerufen, die größer als null ist, geht die Anwendung beim Senden der Antwortnachricht entsprechend einer der nachfolgend aufgeführten Möglichkeiten vor:
   - Die verbleibende Ablaufzeit aus der Anforderungsnachricht in die Antwortnachricht kopieren
   - Die Ablaufzeit wird in der Antwortnachricht auf einen expliziten Wert größer als null gesetzt.
   - Die Ablaufzeit wird in der Antwortnachricht auf MQEI_UNLIMITED gesetzt.

   Welche Aktion durchgeführt wird, hängt von dem Design der Anwendung ab. Die Standardaktion beim Einreihen von Nachrichten in eine Warteschlange für nicht zustellbare Nachrichten muss jedoch der Erhalt der verbleibenden Ablaufzeit der Nachricht und das Fortsetzen ihrer Verringerung sein.
8. Auslösenachrichten werden immer mit MQEI_UNLIMITED erstellt.
9. Eine Nachricht, normalerweise in einer Übertragungswarteschlange, deren *Format*-Name MQFMT_XMIT_Q_HEADER ist, verfügt über einen zweiten Nachrichtendeskriptor innerhalb des MQXQH. Aus diesem Grund sind ihr zwei *Expiry*-Felder zugehörig. In diesem Fall sind die folgenden zusätzlichen Punkte zu beachten:
   - Wenn eine Anwendung eine Nachricht in eine ferne Warteschlange einreiht, fügt der Warteschlangenmanager sie zunächst in eine lokale Übertragungswarteschlange ein und stellt den Anwendungsnachrichtendaten eine MQXQH-Struktur voran. Der Warteschlangenmanager setzt die Werte der zwei *Expiry*-Felder so, dass sie den Angaben der Anwendung entsprechen.

     Wenn eine Anwendung eine Nachricht direkt in eine lokale Übertragungswarteschlange einreiht, müssen die Nachrichtendaten bereits mit einer MQXQH-Struktur beginnen und der Formatname muss MQFMT_XMIT_Q_HEADER sein. In diesem Fall muss die Anwendung die Werte dieser zwei *Expiry*-Felder nicht so wählen, dass sie übereinstimmen. (Der Warteschlangenmanager überprüft, ob das *Expiry*-Feld innerhalb des MQXQH einen gültigen Wert enthält und ob die Nachrichtendaten lang genug sind, um ihn mit einzubeziehen). Bei einer Anwendung, die direkt in die Übertragungswarteschlange schreiben kann, muss die Anwendung einen Header der Übertragungswarteschlange mit dem eingebetteten Nachrichtendeskriptor erstellen. Wenn jedoch der Ablaufwert des Nachrichtendeskriptors, der in die Übertragungswarteschlange geschrieben wird, inkonsistent mit dem Wert in dem eingebetteten Nachrichtendeskriptor ist, kommt es zu einer Ablauffehlerzurückweisung.
   - Wenn von einer Warteschlange, unabhängig davon, ob es sich um eine normale oder eine Übertragungswarteschlange handelt, ein *Format*-Name der Art MQFMT_XMIT_Q_HEADER abgerufen wird, verringert der Warteschlangenmanager *beide Expiry*-Felder um die Zeitspanne, die während des Wartens in der Warteschlange verstrichen ist. Es tritt kein Fehler auf, wenn die Nachrichten nicht lang genug sind, um das *Expiry*-Feld in dem MQXQH einzubeziehen.
   - Der Warteschlangenmanager verwendet das *Expiry*-Feld im gesonderten Nachrichtendeskriptor, also nicht dasjenige des Nachrichtendeskriptors, der in der MQXQH-Struktur eingebettet ist, um zu überprüfen, ob es zulässig ist, die Nachricht zu löschen.
   - Wenn die ursprünglichen Werte der beiden *Expiry*-Felder sich unterscheiden, kann die *Expiry*-Zeit im gesonderten Nachrichtendeskriptor bei Abrufen der Nachricht größer als null sein (was bedeutet, dass es nicht zulässig ist, die Nachricht zu löschen), während die Zeit entsprechend des

Expiry-Feldes im MQXQH abgelaufen ist. In diesem Fall wird das *Expiry*-Feld im MQXQH auf null gesetzt.

10. Die Ablaufzeit einer Antwortnachricht, die von der IMS-Brücke zurückgegeben wird, ist unbegrenzt, es sei denn, das Flags-Feld des MQIIH ist auf MQIIH_PASS_EXPIRATION gesetzt. Weitere Informationen hierzu finden Sie im Abschnitt Flags.

Der folgende Sonderwert wird erkannt:

MQEI_UNLIMITED
Die Nachricht besitzt eine uneingeschränkte Ablaufzeit.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert des Felds ist MQEI_UNLIMITED.

Feedback (MQLONG)
Das Feedbackfeld wird mit einer Nachricht des Typs MQMT_REPORT verwendet, um die Spezifik des Berichts anzugeben, und ist nur mit diesem Typ Nachricht aussagekräftig.

Das Feld kann einen der MQFB_*- oder der MQRC_*-Werte enthalten. Rückmeldungscodes werden wie folgt gruppiert:

MQFB_NONE
Keine Rückmeldung

MQFB_SYSTEM_FIRST
Das ist der niedrigste Wert für vom System erstelltes Feedback.

MQFB_SYSTEM_LAST
Höchster Wert für vom System generierte Rückmeldung

Der Bereich von MQFB_SYSTEM_FIRST bis MQFB_SYSTEM_LAST der vom System erstellten Rückmeldungscodes enthält die allgemeinen Rückmeldungscodes, die in diesem Thema (MQFB_*) aufgeführt sind und außerdem die Ursachencodes (MQRC_*), die auftreten können, wenn die Nachricht nicht in die Zielwarteschlange eingereiht werden kann.

MQFB_APPL_FIRST
Das ist der niedrigste Wert für von der Anwendung erstelltes Feedback.

MQFB_APPL_LAST
Höchster Wert für von der Anwendung generierte Rückmeldung

Anwendungen, die Berichtsnachrichten erstellen, dürfen - abgesehen von MQFB_QUIT - keine Rückmeldungscodes im systemdefinierten Bereich verwenden, es sei denn, sie wollen Berichtsnachrichten simulieren, die vom Warteschlangenmanager oder Nachrichtenkanalagenten erstellt wurden.

Beim MQPUT-Aufruf und beim MQPUT1-Aufruf muss der angegebene Wert entweder MQFB_NONE sein oder innerhalb des vom System oder der Anwendung definierten Bereichs liegen. Dies wird unabhängig vom Wert von *MsgType* überprüft.

Allgemeine Rückkopplungscodes:

MQFB_COA
Bestätigung des Eingangs in der Zielwarteschlange (siehe MQRO_COA).

MQFB_COD
Empfangsbestätigung der empfangenden Anwendung (siehe MQRO_COD).

MQFB_EXPIRATION
Die Nachricht wurde verworfen, da sie nicht aus der Zielwarteschlange entfernt wurde, bevor ihre Ablaufzeit verstrichen war.

MQFB_PAN
Benachrichtigung über eine positive Aktion (siehe MQRO_PAN).

MQFB_NAN
Benachrichtigung über eine negative Aktion (siehe MQRO_NAN).

MQFB_QUIT
Beendigung der Anwendung.

Dieses Feld kann von einem Programm zur Auslastungsplanung verwendet werden, um die Anzahl Instanzen eines Anwendungsprogramms zu steuern, die ausgeführt werden. Wird eine MQMT_REPORT-Nachricht mit diesen Rückmeldungscodes an eine Instanz des Anwendungsprogramms gesendet, so wird der Instanz dadurch angezeigt, dass sie die Verarbeitung beenden soll. Die Einhaltung dieser Konvention ist jedoch nur für die Anwendung von Bedeutung. Sie wird nicht vom Warteschlangenmanager erzwungen.

Kanalrückmeldungscodes:

MQFB_CHANNEL_COMPLETED
Ein Kanal wurde normal beendet.

MQFB_CHANNEL_FAIL
Ein Kanal wurde abnormal beendet und wechselt in den STOPPED-Status.

MQFB_CHANNEL_FAIL_RETRY
Ein Kanal wurde abnormal beendet und wechselt in den RETRY-Status.

Rückmeldungscodes der IMS-Brücke

Diese Codes werden verwendet, wenn ein unerwarteter IMS-OTMA-Prüfcode erhalten wird. Der Prüfcode oder, falls der Prüfcode 0x1A entspricht, der diesem Prüfcode zugeordnete Ursachencode werden im *Feedback* angegeben.

1. Für *Feedback*-Codes im Bereich MQFB_IMS_FIRST (300) bis MQFB_IMS_LAST (399) wurde ein Prüfcode empfangen, der nicht 0x1A entspricht. Der *Prüfcode* wird vom Ausdruck angegeben (*Feedback*: MQFB_IMS_FIRST+1).
2. Bei *Feedback*-Codes im Bereich von MQFB_IMS_NACK_1A_REASON_FIRST (600) bis MQFB_IMS_NACK_1A_REASON_LAST (855) wurde ein Prüfcode empfangen, der 0x1A entspricht. Der dem Prüfcode zugeordnete *Ursachencode* wird vom Ausdruck angegeben (*Feedback*: MQFB_IMS_NACK_1A_REASON_FIRST).

Die Bedeutung der IMS-OTMA-Prüfcodes und der zugehörigen Ursachencodes werden im *Open Transaction Manager Access Guide and Reference* beschrieben.

Die folgenden Rückmeldungscodes können von der IMS-Brücke erstellt werden:

MQFB_DATA_LENGTH_ZERO
Eine Segmentlänge in den Anwendungsdaten der Nachricht ist null.

MQFB_DATA_LENGTH_NEGATIVE
Eine Segmentlänge in den Anwendungsdaten der Nachricht ist negativ.

MQFB_DATA_LENGTH_TOO_BIG
Eine Segmentlänge in den Anwendungsdaten der Nachricht war zu groß.

MQFB_BUFFER_OVERFLOW
Der Wert in einem der Längenfelder führt zum Überlauf der Daten im Nachrichtenpuffer.

MQFB_LENGTH_OFF_BY_ONE
Der Wert eines der Längenfelder war 1 Byte zu kurz.

MQFB_IIH_ERROR
Das *Format*-Feld im MQMD gibt MQFMT_IMS an, aber die Nachricht beginnt nicht mit einer gültigen MQIIH-Struktur.

MQFB_NOT_AUTHORIZED_FOR_IMS
Die Benutzer-ID im Nachrichtendeskriptor MQMD oder das Kennwort im *Authenticator*-Feld in der MQIIH- Struktur hat die Gültigkeitsprüfung durch die IMS-Brücke nicht bestanden. Aus diesem Grund wurde die Nachricht nicht an IMS übermittelt.

MQFB_IMS_ERROR
IMS hat einen unerwarteten Fehler zurückgegeben. Ziehen Sie das WebSphere MQ-Fehlerprotokoll auf dem System, auf dem sich die IMS-Brücke befindet, zu Rate, um weitere Informationen zu diesem Fehler zu erhalten.

MQFB_IMS_FIRST
Wenn der IMS-OTMA-Prüfcode nicht 0x1A ist, befinden sich die IMS-generierten Rückmeldungscodes im Bereich von MQFB_IMS_FIRST (300) bis MQFB_IMS_LAST (399). Der IMS-OTMA-Prüfcode selbst ist *Feedback* minus MQFB_IMS_ERROR.

MQFB_IMS_LAST
Das ist der Höchstwert für IMS-generierte Rückmeldungen, wenn der Prüfcode nicht 0x1A entspricht.

MQFB_IMS_NACK_1A_REASON_FIRST
Wenn der Prüfcode 0x1A ist, befinden sich die IMS-generierten Rückmeldungscodes im Bereich MQFB_IMS_NACK_1A_REASON_FIRST (600) bis MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST
Das ist der Höchstwert für IMS-generierte Rückmeldungen, wenn der Prüfcode 0x1A entspricht.

Rückmeldungscodes der CICS-Brücke: Die nachfolgend aufgeführten Rückmeldungscodes können von der CICS-Brücke erstellt werden:

MQFB_CICS_APPL_ABENDED
Das in der Nachricht angegebene Anwendungsprogramm wurde abnormal beendet. Dieser Rückmeldungscode kommt nur im *Reason*-Feld der MQDLH-Struktur vor.

MQFB_CICS_APPL_NOT_STARTED
Der Befehl "EXEC CICS LINK" für das in der Nachricht angegebene Anwendungsprogramm ist fehlgeschlagen. Dieser Rückmeldungscode kommt nur im *Reason*-Feld der MQDLH-Struktur vor.

MQFB_CICS_BRIDGE_FAILURE
Die CICS-Brücke wurde abnormal beendet, ohne die normale Fehlerbehandlung abzuschließen.

MQFB_CICS_CCSID_ERROR
Zeichensatzkennung nicht gültig

MQFB_CICS_CIH_ERROR
Die CICS-Headerstruktur fehlt oder ist nicht gültig.

MQFB_CICS_COMMAREA_ERROR
Die Länge des CICS-Kommunikationsbereiches ist nicht gültig.

MQFB_CICS_CORREL_ID_ERROR
Die Korrelations-ID ist nicht gültig.

MQFB_CICS_DLQ_ERROR
Die CICS-Brückentask konnte keine Antwort auf diese Anforderung in die Warteschlange für nicht zustellbare Nachrichten kopieren. Die Anforderung wurde zurückgesetzt.

MQFB_CICS_ENCODING_ERROR
Die Codierung ist nicht gültig.

MQFB_CICS_INTERNAL_ERROR
Die CICS-Brücke hat einen unerwarteten Fehler festgestellt.

Dieser Rückmeldungscode kommt nur im *Reason*-Feld der MQDLH-Struktur vor.

MQFB_CICS_NOT_AUTHORIZED
Benutzer-ID nicht berechtigt oder Kennwort nicht gültig

Dieser Rückmeldungscode kommt nur im *Reason*-Feld der MQDLH-Struktur vor.

MQFB_CICS_UOW_BACKED_OUT
Diese Arbeitseinheit wurde aus einem der folgenden Gründe zurückgesetzt:

- Ein Fehler wurde erkannt, während eine andere Anforderung in derselben Arbeitseinheit verarbeitet wurde.
- Ein CICS-Abbruch trat auf, während die Arbeitseinheit in Bearbeitung war.

MQFB_CICS_UOW_ERROR
Das *UOWControl*-Steuerfeld der Arbeitseinheit ist nicht gültig.

Rückmeldungscode der Traceroute-Nachricht:

MQFB_ACTIVITY
Dieser Code wird mit dem MQFMT_EMBEDDED_PCF-Format verwendet, um zu ermöglichen, dass Benutzerdaten Aktivitätenberichten folgen können.

MQFB_MAX_ACTIVITIES
Dieser Code wird zurückgegeben, wenn die Traceroute-Nachricht gelöscht wird, weil die Zahl der Aktivitäten, in die die Nachricht einbezogen war, den maximal erlaubten Grenzwert für Aktivitäten überschreitet.

MQFB_NOT_FORWARDED
Dieser Code wird zurückgegeben, wenn die Traceroute-Nachricht gelöscht wird, weil sie an eine ferne Warteschlange gesendet werden soll, die keine Traceroute-Nachrichten unterstützt.

MQFB_NOT_DELIVERED
Dieser Code wird zurückgegeben, wenn die Traceroute-Nachricht gelöscht wird, weil sie in eine lokale Warteschlange eingereiht werden soll.

MQFB_UNSUPPORTED_FORWARDING
Dieser Code wird zurückgegeben, wenn die Traceroute-Nachricht gelöscht wird, weil ein Wert des weiterleitenden Parameters nicht erkannt wird und sich in der abgelehnten Bitmaske befindet.

MQFB_UNSUPPORTED_DELIVERY
Dieser Code wird zurückgegeben, wenn die Traceroute-Nachricht gelöscht wird, weil ein Wert des bereitstellenden Parameters nicht erkannt wird und sich in der abgelehnten Bitmaske befindet.

Ursachencodes von WebSphere MQ: Bei Ausnahmeberichtsnachrichten enthält *Feedback* einen WebSphere MQ-Ursachencode. Folgende Ursachencodes sind möglich:

MQRC_PUT_INHIBITED
(2051, X'803') Put-Aufrufe sind für die Warteschlange gesperrt.

MQRC_Q_FULL
(2053, X'805') Die Warteschlange enthält bereits die maximale Anzahl von Nachrichten.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') Auf dem Datenträger ist für die Warteschlange kein Speicherplatz verfügbar.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800') Die Warteschlange unterstützt keine persistenten Nachrichten.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Die Nachrichtenlänge überschreitet den maximalen Wert des Warteschlangenmanagers.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Nachrichtenlänge überschreitet den Maximalwert für die Warteschlange.

Eine vollständige Liste von Ursachencodes finden Sie an den nachfolgend aufgeführten Orten:

- Bei WebSphere MQ for z/OS finden Sie sie im Abschnitt API-Ursachencodes.
- Für alle anderen Plattformen finden Sie sie unter API-Beendigungs- und Ursachencodes.

.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQFB_NONE.

Format (MQCHAR8)

Dies ist ein Name, den der Sender der Nachricht verwendet, um dem Empfänger die Datenart in der Nachricht zu melden. Sämtliche Zeichen des Zeichensatzes des Warteschlangenmanagers können für diesen Namen verwendet werden, allerdings müssen Sie den Namen entsprechend der nachfolgenden Bedingungen bilden:

- Großbuchstaben A - Z
- Numerische Ziffern 0 bis 9

Wenn andere Zeichen verwendet werden, ist es eventuell nicht möglich, den Namen vom Zeichensatz des sendenden in den Zeichensatz des empfangenden Warteschlangenmanagers zu übersetzen.

Füllen Sie den Namen bis zur Länge des Felds mit Leerzeichen auf oder verwenden Sie ein Nullzeichen, um den Namen vor dem Ende des Felds zu beenden; null und jegliche nachfolgenden Zeichen werden als Leerzeichen behandelt. Geben Sie keinen Namen mit führenden oder eingebetteten Leerzeichen an. Für den MQGET-Aufruf gibt der Warteschlangenmanager den Namen mit Leerzeichen aufgefüllt bis zur Länge des Felds zurück.

Der Warteschlangenmanager prüft nicht, ob der Name die oben beschriebenen Empfehlungen einhält.

Namen, die MQ in Groß- oder Kleinbuchstaben oder in Groß-/Kleinschreibung beginnen, haben vom Warteschlangenmanager definierte Bedeutungen; verwenden Sie keine Namen, die mit diesen Buchstaben beginnen, für Ihre eigenen Formate.Folgende Formate sind im Warteschlangenmanager integriert:

MQFMT_NONE
Die Art der Daten ist undefiniert: Die Daten können nicht konvertiert werden, wenn die Nachricht von einer Warteschlange unter Verwendung der Option MQGMO_CONVERT abgerufen wird.

Wenn Sie beim MQGET-Aufruf MQGMO_CONVERT angeben und sich der Zeichensatz oder die Codierung der Daten der Nachricht von den Angaben im *MsgDesc*-Parameter unterscheiden, wird die Nachricht (vorausgesetzt, es liegen keine anderen Fehler vor) mit den folgenden Beendigungs- und Ursachencodes zurückgegeben:

- Beendigungscode MQCC_WARNING und Ursachencode MQRC_FORMAT_ERROR, wenn die MQFMT_NONE-Daten am Anfang der Nachricht sind.
- Beendigungscode MQCC_OK und Ursachencode MQRC_NONE, wenn die MQFMT_NONE-Daten am Ende der Nachricht sind (das heißt, wenn davor mindestens eine MQ-Headerstruktur angegeben ist). Die MQ-Headerstrukturen werden in diesem Fall in den angeforderten Zeichensatz und die angeforderte Codierung konvertiert.

Für die Programmiersprache C ist auch die Konstante MQFMT_STRUC_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_STRUC_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_ADMIN
Die Nachricht ist eine Befehlsserveranforderung oder Antwortnachricht im programmierbaren Befehlsformat (PCF). Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird. Weitere Informationen zur Verwendung von Nachrichten im Programmable Command Format finden Sie unter Programmable Command Format verwenden.

Für die Programmiersprache C ist auch die Konstante MQFMT_ADMIN_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_ADMIN, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_CICS
Die Nachrichtendaten beginnen mit dem CICS-Informationsheader MQCIH, gefolgt von den Anwendungsdaten. Der Formatname der Anwendungsdaten wird vom *Format*-Feld in der MQCIH-Struktur angegeben.

Geben Sie unter z/OS die MQGMO_CONVERT-Option beim MQGET-Aufruf an, um Nachrichten mit dem Format MQFMT_CICS zu konvertieren.

Für die Programmiersprache C ist auch die Konstante MQFMT_CICS_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_CICS, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_COMMAND_1
Die Nachricht ist eine MQSC-Befehlsserver-Antwortnachricht, die die Objektzahl, den Beendigungscode und den Ursachencode enthält. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_COMMAND_1_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_COMMAND_1, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_COMMAND_2
Die Nachricht ist eine MQSC-Befehlsserver-Antwortnachricht, die Informationen zu den angeforderten Objekten enthält. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_COMMAND_2_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_COMMAND_2, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_DEAD_LETTER_HEADER
Die Nachrichtendaten beginnen mit dem Header für nicht zustellbare Nachrichten MQDLH. Die Daten der ursprünglichen Nachricht folgen sofort nach der MQDLH-Struktur. Der Formatname der ursprünglichen Nachricht wird von dem *Format*-Feld der MQDLH-Struktur angegeben; weitere Informationen zu dieser Struktur finden Sie im Abschnitt „MQDLH - Header für nicht zustellbare Nachrichten” auf Seite 329. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Berichte mit Bestätigung bei Eingang und Berichte mit Bestätigung bei Zustellung werden für Nachrichten mit dem *Format* QFMT_DEAD_LETTER_HEADER nicht generiert.

Für die Programmiersprache C ist auch die Konstante MQFMT_DEAD_LETTER_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_DEAD_LETTER_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_DIST_HEADER
Die Nachrichtendaten beginnen mit dem Verteilerlistenheader MQDH; dies schließt die Gruppen von MQOR- und MQPMR-Datensätzen ein. Dem Verteilerlistenheader können zusätzliche Daten folgen. Das Format der zusätzlichen Daten (falls solche Daten vorliegen sollten) wird vom *Format*-Feld in der MQDLH-Struktur angegeben; weitere Informationen zu dieser Struktur finden Sie im Abschnitt „MQDH - Verteilerheader” auf Seite 322. Nachrichten mit dem Format MQFMT_DIST_HEADER können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Dieses Format wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Für die Programmiersprache C ist auch die Konstante MQFMT_DIST_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_DIST_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_EMBEDDED_PCF
Format für eine Traceroute-Nachricht, unter der Voraussetzung, dass der Wert des PCF-Befehls auf MQCMD_TRACE_ROUTE gesetzt ist. Mithilfe dieses Formats können Benutzerdaten zusammen mit der Traceroute-Nachricht gesendet werden, vorausgesetzt, dass ihre Anwendungen mit vorausgehenden PCF-Parametern umgehen können.

Der PCF-Header **muss** der erste Header sein, sonst wird die Nachricht nicht wie eine Traceroute-Nachricht behandelt. Das bedeutet, dass die Nachricht nicht Teil einer Gruppe sein kann und dass Traceroute-Nachrichten nicht segmentiert werden können. Wenn eine Traceroute-Nachricht als Teil einer Gruppe gesendet wird, wird die Nachricht mit dem Ursachencode MQRC_MSG_NOT_ALLOWED_IN_GROUP zurückgewiesen.

Beachten Sie, dass MQFMT_ADMIN auch für das Format einer Traceroute-Nachricht verwendet werden kann, allerdings können in diesem Fall zusammen mit der Traceroute-Nachricht keine Benutzerdaten gesendet werden.

MQFMT_EVENT
Dies ist eine MQ-Ereignisnachricht, die ein aufgetretenes Ereignis meldet. Ereignisnachrichten haben dieselbe Struktur wie programmierbare Befehle. Weitere Informationen zu dieser Struktur finden Sie im Abschnitt PCF-Befehlsnachrichten, Informationen zu Ereignissen finden Sie im Abschnitt Ereignisüberwachung.

Version-1-Ereignisnachrichten können in allen Umgebungen konvertiert werden, wenn die MQGMO_CONVERT-Option im MQGET-Aufruf angegeben ist. Ereignisnachrichten Version-2 können nur unter z/OS konvertiert werden.

Für die Programmiersprache C ist auch die Konstante MQFMT_EVENT_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_EVENT, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_IMS
Die Nachrichtendaten beginnen mit dem IMS-Informationsheader MQIIH, gefolgt von den Anwendungsdaten. Der Formatname der Anwendungsdaten wird vom *Format*-Feld in der MQCIIH-Struktur angegeben.

Einzelheiten dazu, wie die MQIIH-Struktur bei Verwenden von MQGET mit MQGMO_CONVERT verarbeitet wird, finden Sie in den Abschnitten „Format (MQCHAR8)" auf Seite 385 und „ReplyToFormat (MQCHAR8)" auf Seite 385.

Für die Programmiersprache C ist auch die Konstante MQFMT_IMS_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_IMS, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_IMS_VAR_STRING
Die Nachricht ist eine IMS-Variablenzeichenfolge, also eine Zeichenfolge mit dem Format `llzzccc`, wobei:

`ll`
Ein 2-Byte-Längenfeld, das die Gesamtlänge des IMS-Variablenzeichenfolgeelements angibt. Diese Länge entspricht der Länge von `ll` (2 Byte) plus der Länge von `zz` (2 Byte) plus der Länge der Zeichenfolge selbst. `ll` ist eine binäre Ganzzahl (2 Byte) der Codierung, die vom *Encoding*-Feld angegeben wird.

`zz`
Ein Feld mit einer Länge von 2 Byte; es enthält Flags, die für IMS relevant sind. `zz` ist eine Bytefolge von zwei MQBYTE-Feldern und wird ohne Änderung vom Sender an den Empfänger übertragen, das heißt, `zz` wird in keiner Weise umgewandelt.

`ccc`
ist eine Zeichenfolge variabler Länge, die `ll-4` Zeichen umfasst. `ccc` wird im Zeichensatz vom *CodedCharSetId*-Feld angegeben.

Bei z/OS können die Nachrichtendaten aus einer Folge von zusammengesetzten IMS-Variablenzeichenfolgen bestehen, wobei jede der Zeichenfolgen dem Format `llzzccc` entspricht. Zwischen aufeinanderfolgenden IMS-Variablenzeichenfolgen darf es keine übersprungenen Bytes geben. Das bedeutet, dass, wenn die erste Zeichenfolge eine ungerade Länge aufweist, die zweite Zeichenfolge verschoben sein wird, das heißt, sie wird nicht bei einem Grenzwert beginnen, der ein Vielfaches von zwei ist. Achten Sie darauf, wenn sie solche Zeichenfolgen auf Systemen erstellen, bei denen die Elementardatentypen ausgerichtet sein müssen.

Verwenden Sie die MQGMO_CONVERT-Option beim MQGET-Aufruf, um Nachrichten mit dem Format MQFMT_IMS_VAR_STRING zu konvertieren.

Für die Programmiersprache C ist auch die Konstante MQFMT_IMS_VAR_STRING_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_IMS_VAR_STRING, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_MD_EXTENSION
Die Nachrichtendaten beginnen mit der Nachrichtendeskriptorerweiterung MQMDE, auf die optional weitere Daten folgen (normalerweise die Anwendungsnachrichtendaten). Formatname, Zeichensatz

und Codierung der auf die MQMDE folgenden Daten werden von den *Format*-, *CodedCharSetId*- und *Encoding*-Feldern in der MQMDE angegeben. Weitere Informationen zu dieser Struktur finden Sie unter „MQMDE – Nachrichtendeskriptorerweiterung" auf Seite 453. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_MD_EXTENSION_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_MD_EXTENSION, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_PCF
Die Nachricht ist eine benutzerdefinierte Nachricht, die der Struktur einer Nachricht im Programmable Command Format (PCF) entspricht. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird. Weitere Informationen zur Verwendung von Nachrichten im Programmable Command Format finden Sie unter Programmable Command Format verwenden.

Für die Programmiersprache C ist auch die Konstante MQFMT_PCF_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_PCF, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_REF_MSG_HEADER
Die Nachrichtendaten beginnen mit dem Referenznachrichtenheader MQRMH, auf den optional weitere Daten folgen. Formatname, Zeichensatz und Codierung der Daten werden von den *Format*-, *CodedCharSetId*- und *Encoding*-Feldern in dem MQRMH angegeben. Weitere Informationen zu dieser Struktur finden Sie unter „MQRMH - Header für Referenznachrichten" auf Seite 534. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Dieses Format wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Für die Programmiersprache C ist auch die Konstante MQFMT_REF_MSG_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_REF_MSG_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_RF_HEADER
Die Nachrichtendaten beginnen mit dem Regel- und Formatierungsheader MQRFH, auf die optional weitere Daten folgen. Formatname, Zeichensatz und Codierung der Daten (falls vorhanden) werden von den *Format*-, *CodedCharSetId*- und *Encoding*-Feldern in dem MQRFH angegeben. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_RF_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_RF_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_RF_HEADER_2
Die Nachrichtendaten beginnen mit dem Regel- und Formatierungsheader MQRFH2 der Version 2, auf die optional weitere Daten folgen. Formatname, Zeichensatz und Codierung der optionalen Daten (falls vorhanden) werden von den *Format*-, *CodedCharSetId*- und *Encoding*-Feldern in dem MQRFH2 angegeben. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_RF_HEADER_2_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_RF_HEADER_2, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_STRING
Bei den Anwendungsnachrichtendaten kann es sich entweder um eine SBCS-Zeichenfolge (Einzelbytezeichensatz) oder um eine DBCS-Zeichenfolge (Doppelbytezeichensatz) handeln. Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_STRING_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_STRING, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_TRIGGER
Die Nachricht ist eine Auslösenachricht, die von der MQTM-Struktur beschrieben wird (weitere Informationen zu dieser Struktur finden Sie unter „MQTM - Auslösenachricht" auf Seite 589). Nachrichten dieses Formats können konvertiert werden, wenn die MQGMO_CONVERT-Option beim MQGET-Aufruf angegeben wird.

Für die Programmiersprache C ist auch die Konstante MQFMT_TRIGGER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_TRIGGER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_WORK_INFO_HEADER
Die Nachrichtendaten beginnen mit dem Auslastungsheader MQWIH, auf den die Anwendungsdaten folgen. Der Formatname der Anwendungsdaten wird vom *Format*-Feld in der MQWIH-Struktur angegeben.

Geben Sie bei z/OS beim MQGET-Aufruf die Option MQGMO_CONVERT an, um *Benutzerdaten* in Nachrichten mit dem Format MQFMT_WORK_INFO_HEADER zu konvertieren. Allerdings wird die MQWIH-Struktur selbst immer mit dem Zeichensatz und der Codierung des Warteschlangenmanagers zurückgegeben (das heißt, dass die MQWIH-Struktur konvertiert wird, unabhängig davon, ob die MQGMO_CONVERT-Option angegeben ist).

Für die Programmiersprache C ist auch die Konstante MQFMT_WORK_INFO_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_WORK_INFO_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

MQFMT_XMIT_Q_HEADER
Die Nachrichtendaten beginnen mit dem Header der Übertragungswarteschlange MQXQH. Die Daten der ursprünglichen Nachricht folgen direkt auf die MQXQH-Struktur. Der Formatname der ursprünglichen Nachricht wird von dem *Format*-Feld in der MQMD-Struktur angegeben, das Teil des Headers der Übertragungswarteschlange MQXQH ist. Weitere Informationen zu dieser Struktur finden Sie unter „MQXQH – Header der Übertragungswarteschlange" auf Seite 609.

Berichte mit Bestätigung bei Eingang und Berichte mit Bestätigung bei Zustellung werden für Nachrichten mit dem *Format* MQFMT_XMIT_Q_HEADER nicht generiert.

Für die Programmiersprache C ist auch die Konstante MQFMT_XMIT_Q_HEADER_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQFMT_XMIT_Q_HEADER, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

GroupId (MQBYTE24)

Dies ist eine Bytefolge, die verwendet wird, um die Nachrichtengruppe oder logische Nachricht anzugeben, zu der die physische Nachricht gehört. *GroupId* wird auch verwendet, wenn die Segmentierung der Nachricht zulässig ist. In all diesen Fällen hat *GroupId* einen Wert ungleich null und im *MsgFlags*-Feld ist mindestens eines der folgenden Flags gesetzt:

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

Wenn keines dieser Flags gesetzt ist, hat *GroupId* den Spezialnullwert MQGI_NONE.

In folgenden Fällen muss die Anwendung dieses Feld beim MQPUT- oder MQGET-Aufruf nicht setzen:

- Beim MQPUT-Aufruf ist MQPMO_LOGICAL_ORDER angegeben.
- Beim MQGET-Aufruf wird MQMO_MATCH_GROUP_ID *nicht* angegeben.

Dies sind die empfohlenen Vorgehensweisen für die Verwendung dieser Aufrufe bei Nachrichten, die keine Berichtsnachrichten sind. Wenn jedoch die Anwendung weitergehend gesteuert werden muss oder es sich bei dem Aufruf um einen MQPUT1-Aufruf handelt, muss die Anwendung sicherstellen, dass *GroupId* auf einen geeigneten Wert gesetzt wurde.

Nachrichtengruppen und Segmente können nur dann ordnungsgemäß verarbeitet werden, wenn die Gruppen-ID eindeutig ist. Aus diesem Grund *dürfen Anwendungen nicht selbst ihre Gruppen-IDs erstellen*, sondern müssen sich stattdessen an folgende Verfahrensweisen halten:

- Wenn MQPMO_LOGICAL_ORDER angegeben wird, generiert der Warteschlangenmanager automatisch für die erste Nachricht der Gruppe oder für das erste Segment einer logischen Nachricht eine eindeutige Gruppen-ID und verwendet diese Gruppen-ID für die übrigen Nachrichten der Gruppe bzw. Segmente der logischen Nachricht, sodass die Anwendung keine speziellen Maßnahmen ergreifen muss. Dies ist die empfohlene Vorgehensweise.
- Wird MQPMO_LOGICAL_ORDER *nicht* angegeben, muss die Anwendung beim Warteschlangenmanager die Erstellung einer Gruppen-ID anfordern, indem sie beim ersten MQPUT- oder MQPUT1-Aufruf einer Nachricht in der Gruppe oder dem Segment der logischen Nachricht *GroupId* auf MQGI_NONE setzt. Die vom Warteschlangenmanager infolge dieses Aufrufs zurückgegebene Ausgabe muss dann für die verbleibenden Nachrichten der Gruppe oder der Segmente der logischen Nachricht verwendet werden. Wenn eine Nachrichtengruppe segmentierte Nachrichten enthält, muss die gleiche Gruppen-ID für alle Segmente und Nachrichten der Gruppe verwendet werden.

  Wenn MQPMO_LOGICAL_ORDER nicht angegeben wird, können Nachrichten in Gruppen bzw. Segmente logischer Nachrichten in beliebiger Reihenfolge eingereiht werden (beispielsweise in umgekehrter Reihenfolge), allerdings muss die Gruppen-ID von dem *ersten* MQPUT- oder MQPUT1-Aufruf, der für eine beliebige Nachricht ausgegeben wird, eingereiht werden.

Bei Eingabe mit dem MQPUT-oder dem MQPUT1-Aufruf verwendet der Warteschlangenmanager den Wert, der in Physische Reihenfolge in einer Warteschlange beschrieben wird. Bei Eingabe in MQPUT- und MQPUT1-Aufrufe setzt der Warteschlangenmanager dieses Feld auf den Wert, der mit der Nachricht übermittelt wurde, falls es sich bei dem geöffneten Objekt um eine einzelne Warteschlange und nicht um eine Verteilerliste handelt, lässt den Wert aber unverändert, falls das geöffnete Objekt eine Verteilerliste ist. Falls die Anwendung über die erstellten Gruppen-IDs informiert sein muss, muss die Anwendung im letzteren Fall MQPMR-Datensätze mit dem *GroupId*-Feld bereitstellen.

Bei Eingabe mit dem MQGET-Aufruf verwendet der Warteschlangenmanager den in Tabelle 506 auf Seite 369 beschriebenen Wert. Bei Ausgabe des MQGET-Aufrufs setzt der Warteschlangenmanager dieses Feld auf den Wert für die abgerufene Nachricht.

Der folgende spezielle Wert ist definiert:

MQGI_NONE
Keine Gruppen-ID angegeben

Der Wert ist eine binäre Null für die Feldlänge. Dies ist der Wert, der für Nachrichten verwendet wird, die sich nicht Gruppen oder Segmenten logischer Nachrichten befinden und für die Segmentierung nicht zulässig ist.

Für die Programmiersprache C ist auch die Konstante MQGI_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQGI_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_GROUP_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQGI_NONE. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQMD_VERSION_2 ist.

MsgFlags (MQLONG)
MsgFlags sind Flags, die Attribute der Nachricht angeben oder die Verarbeitung der Nachricht steuern.

MsgFlags können den folgenden Kategorien entsprechen:

- Segmentierungsflags

- Statusflags

Segmentierungsflags: Ist eine Nachricht zu groß für eine Warteschlange, schlägt der Versuch, die Nachricht in die Warteschlange einzureihen, normalerweise fehl. Segmentierung ist ein Verfahren, bei dem der Warteschlangenmanager oder die Anwendung die Nachricht in kleinere, Segmente genannte Teile aufteilt und jedes Segment als eine gesonderte physische Nachricht in die Warteschlange einreiht. Die Anwendung, die die Nachricht abruft, kann die Segmente entweder nacheinander abrufen oder beim Warteschlangenmanager anfordern, die Segmente wieder zu einer einzigen Nachricht zusammenzufügen, die vom MQGET-Aufruf zurückgegeben wird. Letzteres wird erreicht, indem die Option MQGMO_COMPLETE_MSG beim MQGET-Aufruf angegeben und ein Puffer zur Verfügung gestellt wird, der ausreicht, um die gesamte Nachricht aufzunehmen. (Weitere Informationen zur Option MQGMO_COMPLETE_MSG finden Sie im Abschnitt „MQGMO – Nachrichtenabrufoptionen" auf Seite 348.) Eine Nachricht kann beim sendenden, bei einem temporären oder beim Ziel-Warteschlangenmanager segmentiert werden.

Für die Segmentierung einer Nachricht können Sie eine der folgenden Optionen angeben:

MQMF_SEGMENTATION_INHIBITED
Diese Option verhindert, dass die Nachricht vom Warteschlangenmanager in Segmente aufgeteilt wird. Wenn sie für eine Nachricht angegeben wird, die bereits ein Segment ist, verhindert diese Option, dass das Segment in noch kleinere Segmente aufgeteilt wird.

Der Wert dieses Flags ist auf eine binäre Null gesetzt. Dies ist die Standardeinstellung.

MQMF_SEGMENTATION_ALLOWED
Diese Option ermöglicht, dass die Nachricht vom Warteschlangenmanager in Segmente aufgeteilt wird. Wenn sie für eine Nachricht angegeben wird, die bereits ein Segment ist, ermöglicht diese Option, dass das Segment in noch kleinere Segmente aufgeteilt wird. MQMF_SEGMENTATION_ALLOWED kann angegeben werden, ohne dass MQMF_SEGMENT oder QMF_LAST_SEGMENT angegeben werden.

- Bei z/OS unterstützt der Warteschlangenmanager die Segmentierung von Nachrichten nicht. Wenn eine Nachricht zu groß für die Warteschlange ist, schlägt der MQPUT- bzw. MQPUT1-Aufruf mit dem Ursachencode MQRC_MSG_TOO_BIG_FOR_Q fehl. Dennoch kann die Option MQMF_SEGMENTATION_ALLOWED immer noch angegeben werden, sodass es möglich wird, die Nachricht bei einem fernen Warteschlangenmanager zu segmentieren.

Wenn der Warteschlangenmanager eine Nachricht segmentiert, aktiviert er das Flag MQMF_SEGMENT in der Kopie des MQMD, der mit jedem Segment verschickt wird. Aber die Einstellungen dieser Flags in den MQMD, die die Anwendung bei den MQPUT- und MQPUT1-Aufrufen zur Verfügung stellt, werden nicht verändert. Beim letzten Segment der logischen Nachricht aktiviert der Warteschlangenmanager außerdem im mit dem Segment gesendeten MQMD das Flag MQMF_LAST_SEGMENT.

Anmerkung: Seien Sie vorsichtig, wenn Sie Nachrichten einreihen, bei denen MQMF_SEGMENTATION_ALLOWED, aber nicht MQPMO_LOGICAL_ORDER angegeben ist. Wenn die Nachricht:

- kein Segment ist und
- nicht Teil einer Gruppe und
- nicht weitergeleitet wird,

dann muss die Anwendung vor *jedem* MQPUT- und MQPUT1-Aufruf das *GroupId*-Feld auf MQGI_NONE zurücksetzen, damit der Warteschlangenmanager für jede Nachricht eine eindeutige Gruppen-ID erstellen kann. Wird dies nicht getan, können mit der betroffenen Nachricht nicht in Beziehung stehende Nachrichten dieselbe Gruppen-ID besitzen, wodurch es im Anschluss zu einer falschen Verarbeitung kommen kann. Weitere Informationen dazu, wann das *GroupId*-Feld zurückgesetzt werden sollte, finden Sie in den Beschreibungen des *GroupId*-Felds und der Option MQPMO_LOGICAL_ORDER.

Der Warteschlangenmanager teilt Nachrichten nach Bedarf in Segmente auf, so dass die Segmente (einschließlich der benötigten Headerdaten) passend in die Warteschlange eingereiht werden können. Allerdings gibt es eine Untergrenze für die Größe eines vom Warteschlangenmanager erstellten Segments und nur das zuletzt erstellte Segment einer Nachricht kann diesen Grenzwert unterschreiten (die Untergrenze für die Größe eines von einer Anwendung generierten Segments ist ein Byte).

Vom Warteschlangenmanager erstellte Segmente könnten unterschiedliche Längen haben. Der Warteschlangenmanager verarbeitet die Nachricht wie folgt:

- Benutzerdefinierte Formate werden mit Grenzwerten aufgeteilt, die jeweils ein Vielfaches von 16 Bytes betragen; der Warteschlangenmanager erstellt (abgesehen vom letzten Segment) keine Segmente, die kleiner sind als 16 Bytes.
- Integrierte Formate - außer MQFMT_STRING - werden an der Spezifik der vorhandenen Daten entsprechenden Stellen aufgeteilt. Allerdings teilt der Warteschlangenmanager eine Nachricht niemals in der Mitte einer WebSphere MQ-Headerstruktur auf. Das bedeutet, dass ein Segment, das eine einzelne MQ-Headerstruktur enthält, nicht weiter vom Warteschlangenmanager aufgeteilt werden kann, und führt dazu, dass die kleinstmögliche Segmentgröße für die entsprechende Nachricht größer als 16 Bytes ist.

  Das zweite oder später vom Warteschlangenmanager generierte Segment beginnt mit einem der Folgenden:

  – Einer MQ-Headerstruktur
  – Dem Anfang der Anwendungsnachrichtendaten
  – Einem Teil der Nachrichtendaten der Anwendung
- MQFMT_STRING wird ohne Berücksichtigung der Spezifik der Daten (SBCS, DBCS oder SBCS/DBCS gemischt) aufgeteilt. Wenn die Zeichenfolge DBCS oder SBCS/DBCS gemischt ist, führt dies eventuell zu Segmenten, die nicht von einem Zeichensatz in einen anderen konvertiert werden können. Der Warteschlangenmanager teilt MQFMT_STRING-Nachrichten niemals in Segmente auf, die kleiner sind als 16 Byte. Eine Ausnahme bildet das letzte Segment.
- Der Warteschlangenmanager setzt die Felder *Format*, *CodedCharSetId* und *Encoding* im MQMD der einzelnen Segmente, um die vorhandenen Daten am *Anfang* des Segments korrekt zu beschreiben; bei dem Formatnamen handelt es sich um den Namen eines integrierten oder eines benutzerdefinierten Formats.
- Das *Report*-Feld im MQMD von Segmenten mit einem *Offset* größer als null wird geändert. Bei jedem Berichtstyp wird, wenn die Berichtsoption MQRO_*_WITH_DATA ist, das Segment aber die ersten 100 Bytes der Benutzerdaten, also die eventuell vorliegenden WebSphere MQ-Headerstrukturen, nicht beinhalten kann, die Berichtsoption zu MQRO_* geändert.

Der Warteschlangenmanager befolgt die oben genannten Regeln, teilt die Nachrichten aber darüber hinaus in nicht vorhersehbarer Weise auf; strengen Sie keine Vermutung darüber an, an welcher Stelle genau eine Nachricht aufgeteilt wurde.

Bei *persistenten* Nachrichten kann der Warteschlangenmanager nur innerhalb einer Arbeitseinheit segmentieren:

- Wird der MQPUT- oder MQPUT1-Aufruf innerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt, wird diese Arbeitseinheit verwendet. Falls der Aufruf während des Segmentierungsprozesses fehlschlägt, entfernt der Warteschlangenmanager alle Segmente, die aufgrund des fehlgeschlagenen Aufrufs in die Warteschlange eingereiht wurden. Allerdings verhindert das Fehlschlagen nicht, dass die Arbeitseinheit erfolgreich festgeschrieben wird.
- Wenn der Aufruf außerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt wird und keine benutzerdefinierte Arbeitseinheit vorhanden ist, erstellt der Warteschlangenmanager für die Dauer des Aufrufs eine eigene Arbeitseinheit. Wenn der Aufruf erfolgreich ist, schreibt der Warteschlangenmanager die Arbeitseinheit automatisch fest. Wenn der Aufruf fehlschlägt, setzt der Warteschlangenmanager die Arbeitseinheit zurück.
- Falls der Aufruf außerhalb einer benutzerdefinierten Arbeitseinheit ausgeführt wird, eine benutzerdefinierte Arbeitseinheit jedoch vorhanden ist, kann der Warteschlangenmanager keine Segmentierung durchführen. Sollte die Nachricht keine Segmentierung erfordern, kann der Aufruf immer noch erfolgreich ausgeführt werden. Aber falls für den Aufruf eine Segmentierung erforderlich sein sollte, schlägt der Aufruf mit dem Ursachencode MQRC_UOW_NOT_AVAILABLE fehl.

Bei *nicht persistenten* Nachrichten benötigt der Warteschlangenmanager keine Arbeitseinheit, um die Segmentierung durchzuführen.

Seien Sie besonders vorsichtig, wenn Sie Daten von Nachrichten konvertieren, die eventuell segmentiert werden:

- Wenn die empfangende Anwendung beim MQGET-Aufruf Daten konvertiert und die MQGMO_COMPLETE_MSG-Option angibt, übergibt der Datenkonvertierungsexit die vollständige Nachricht, damit sie für den Exit konvertiert werden kann, wobei die Segmentierung der Nachricht für den Exit offensichtlich ist.
- Wenn die empfangende Anwendung ein Segment nach dem anderen abruft, wird der Datenkonvertierungsexit aufgerufen, um ein Segment nach dem anderen zu konvertieren. Der Exit muss deswegen die Daten in einem Segment unabhängig von den Daten der anderen Segmente konvertieren.

  Wenn die Daten so angelegt sind, dass eine willkürliche Segmentierung der Daten unter Berücksichtigung der 16-Byte-Grenze zu Segmenten führt, die vom Exit nicht konvertiert werden können, oder wenn das Format MQFMT_STRING und der Zeichensatz auf DBCS oder SBCS/DBCS gemischt gesetzt ist, muss die sendende Anwendung die Segmente erstellen und einreihen und MQMF_SEGMENTATION_INHIBITED angeben, um eine weitere Segmentierung zu verhindern. Auf diese Weise kann die sendende Anwendung sicherstellen, dass jedes Segment hinreichende Informationen enthält, um dem Datenkonvertierungsexit zu ermöglichen, das Segment erfolgreich zu konvertieren.
- Wenn für einen sendenden Nachrichtenkanalagenten (MCA) Senderkonvertierung angegeben wird, konvertiert der Nachrichtenkanalagent nur Nachrichten, die nicht Segmente logischer Nachrichten sind; der Nachrichtenkanalagent versucht niemals, Nachrichten zu konvertieren, die Segmente sind.

Dieses Flag ist ein Eingabeflag bei MQPUT- und MQPUT1-Aufrufen und ein Ausgabeflag bei MQGET-Aufrufen. Beim letzteren Aufruf meldet der Warteschlangenmanager dem *Segmentation*-Feld in MQGMO auch den Wert des Flags.

Der Anfangswert des Felds ist MQMF_SEGMENTATION_INHIBITED.

Statusflags: Hierbei handelt es sich um Flags, die angeben, ob die physische Nachricht zu einer Nachrichtengruppe gehört, ein Segment einer logischen Nachricht ist oder beiden bzw. keiner dieser Möglichkeiten entspricht. Mindestens eine der folgenden Optionen kann beim MQPUT- und MQPUT1-Aufruf angegeben oder vom MQGET-Aufruf zurückgegeben werden:

MQMF_MSG_IN_GROUP
Die Nachricht ist Mitglied einer Gruppe.

MQMF_LAST_MSG_IN_GROUP
Nachricht ist die letzte logische Nachricht einer Gruppe

Wenn dieses Flag gesetzt ist, aktiviert der Warteschlangenmanager MQMF_MSG_IN_GROUP in der Kopie des MQMD, der mit der Nachricht gesendet wird. Aber die Einstellungen dieser Flags in den MQMD, die die Anwendung bei den MQPUT- und MQPUT1-Aufrufen zur Verfügung stellt, werden nicht verändert.

Eine Gruppe kann aus nur einer logischen Nachricht bestehen. Sollte dies der Fall sein, wird MQMF_LAST_MSG_IN_GROUP gesetzt, das *MsgSeqNumber*-Feld hat allerdings den Wert eins.

MQMF_SEGMENT
Nachricht ist Segment einer logischen Nachricht

Wenn MQMF_SEGMENT ohne MQMF_LAST_SEGMENT angegeben wird, muss die Länge der Anwendungsnachrichtendaten des Segments (wobei die Längen eventuell vorhandener WebSphere MQ-Headerstrukturen *nicht* mit berücksichtigt werden) mindestens eins betragen. Ist die Länge null, schlägt der MQPUT- oder der MQPUT1-Aufruf mit dem Ursachencode MQRC_SEGMENT_LENGTH_ZERO fehl.

Bei z/OS wird diese Option nicht unterstützt, wenn die Nachricht in eine Warteschlange mit dem Indextyp MQIT_GROUP_ID eingereiht wird.

MQMF_LAST_SEGMENT
Nachricht ist das letzte Segment einer logischen Nachricht

Wenn dieses Flag gesetzt ist, aktiviert der Warteschlangenmanager MQMF_SEGMENT in der Kopie des MQMD, der mit der Nachricht gesendet wird. Aber die Einstellungen dieser Flags in den MQMD, die die Anwendung bei den MQPUT- und MQPUT1-Aufrufen zur Verfügung stellt, werden nicht verändert.

Eine logische Nachricht kann aus nur einem Segment bestehen. Sollte dies der Fall sein, wird MQMF_LAST_SEGMENT gesetzt, das *Offset*-Feld hat allerdings den Wert null.

Wenn MQMF_LAST_SEGMENT angegeben wird, kann die Länge der Anwendungsnachrichtendaten im Segment (wobei die Längen eventuell vorhandener WebSphere MQ-Headerstrukturen *nicht* mit berücksichtigt werden) null sein.

Bei z/OS wird diese Option nicht unterstützt, wenn die Nachricht in eine Warteschlange mit dem Indextyp MQIT_GROUP_ID eingereiht wird.

Die Anwendung muss beim Einreihen von Nachrichten sicherstellen, dass diese Flags ordnungsgemäß gesetzt sind. Wenn MQPMO_LOGICAL_ORDER angegeben wird oder beim vorangehenden MQPUT-Aufruf für die Warteschlangenkennung angegeben wurde, müssen die Einstellungen der Flags konsistent mit den vom Warteschlangenmanager für die Warteschlangenkennung beibehaltenen Gruppen- und Segmentinformationen sein. Die folgenden Bedingungen werden bei *aufeinanderfolgenden* MQPUT-Aufrufen für die Warteschlangenkennung angewandt, wenn MQPMO_LOGICAL_ORDER angegeben ist:

- Liegt keine aktuelle Gruppe oder logische Nachricht vor, sind alle diese Flags (und Kombinationen daraus) gültig.
- Sobald MQMF_MSG_IN_GROUP angegeben wurde, muss dieses Flag bestehen bleiben, bis MQMF_LAST_MSG_IN_GROUP angegeben ist. Der Aufruf schlägt mit Ursachencode MQRC_INCOMPLETE_GROUP fehl, wenn dieser Bedingung nicht entsprochen wird.
- Sobald MQMF_SEGMENT angegeben wurde, muss dieses Flag bestehen bleiben, bis MQMF_LAST_SEGMENT angegeben ist. Der Aufruf schlägt mit Ursachencode MQRC_INCOMPLETE_MSG fehl, wenn dieser Bedingung nicht entsprochen wird.
- Sobald MQMF_SEGMENT ohne MQMF_MSG_IN_GROUP angegeben wurde, muss MQMF_MSG_IN_GROUP *deaktiviert* bleiben, und kann erst aktiviert werden, nachdem MQMF_LAST_SEGMENT angegeben wurde. Der Aufruf schlägt mit Ursachencode MQRC_INCOMPLETE_MSG fehl, wenn dieser Bedingung nicht entsprochen wird.

Physische Reihenfolge in einer Warteschlange zeigt die gültigen Flagkombinationen und die Werte, die für die verschiedenen Felder verwendet werden.

Diese Flags sind Eingabeflags bei MQPUT- und MQPUT1-Aufrufen und Ausgabeflags beim MQGET-Aufruf. Beim letzteren Aufruf meldet der Warteschlangenmanager dem *GroupStatus*- und dem *SegmentStatus*-Feld in MQGMO auch den Wert der Flags.

Sie können gruppierte und segmentierte Nachrichten nicht mit Publish/Subscribe verwenden.

Standardflags: Mit den folgende Angaben kann angezeigt werden, dass die Nachricht über Standardattribute verfügt:

MQMF_NONE
Keine Nachrichtenflags (Standardnachrichtenattribute)

Damit wird die Segmentierung blockiert und angezeigt, dass die Nachricht keiner Gruppe angehört und nicht Segment einer logischen Nachricht ist. MQMF_NONE dient zur Unterstützung der Programmdokumentation. Dieses Flag ist nicht zur Verwendung mit einem anderen Flag gedacht, da es jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

Das *MsgFlags*-Feld wird in Unterfelder untergliedert; weitere Informationen finden Sie im Abschnitt „Report options and message flags” auf Seite 904.

Der Anfangswert dieses Felds ist MQMF_NONE. Dieses Feld wird ignoriert, wenn die *Version* kleiner ist als MQMD_VERSION_2.

MsgId (MQBYTE24)

Dies ist eine Bytefolge, die verwendet wird, um Nachrichten voneinander zu unterscheiden. Im Allgemeinen dürfen verschiedene Nachrichten nicht dieselbe Nachrichten-ID haben, obwohl dies vom Warteschlangenmanager zugelassen wird. Die Nachrichten-ID ist eine persistente Eigenschaft der Nachricht, die auch bei einem Neustart des Warteschlangenmanagers bestehen bleibt. Da die Nachrichten-ID eine

Bytefolge und keine Zeichenfolge ist, wird die Nachrichten-ID *nicht* in einen anderen Zeichensatz konvertiert, wenn die Nachricht von einem Warteschlangenmanager zum nächsten übertragen wird.

Wenn für die MQPUT-und MQPUT1-Aufrufe MQMI_NONE oder MQPMO_NEW_MSG_ID von der Anwendung angegeben wird, generiert der Warteschlangenmanager eine eindeutige Nachrichtenkennung.[2]wenn die Nachricht eingereiht wird, und die Anwendung fügt die ID in den mit der Nachricht gesendeten Nachrichtendeskriptor ein. Der Warteschlangenmanager gibt auch diese Nachrichten-ID in dem Nachrichtendeskriptor zurück, der zur sendenden Anwendung gehört. Die Anwendung kann diesen Wert verwenden, um Informationen über bestimmte Nachrichten aufzuzeichnen und auf Abfragen von anderen Komponenten der Anwendung zu reagieren.

Wenn die Nachricht zu einem Thema eingereiht wird, erstellt der Warteschlangenmanager die für veröffentlichte Nachrichten erforderliche eindeutige Nachrichten-ID. Wenn von der Anwendung MQPMO_NEW_MSG_ID angegeben wird, generiert der Warteschlangenmanager eine eindeutige Nachrichten-ID, um sie bei Ausgabe zurückzugeben. Wenn MQMI_NONE von der Anwendung angegeben wird, bleibt der Wert des Felds *MsgId* im MQMD bei der Rückgabe des Aufrufs unverändert.

Weitere Informationen über ständige Veröffentlichungen finden Sie in der Beschreibung von MQPMO_RETAIN im Abschnitt „MQPMO-Optionen (MQLONG)" auf Seite 491.

Wenn die Nachricht in eine Verteilerliste eingereiht wird, generiert der Warteschlangenmanager nach Bedarf eindeutige Nachrichten-IDs, der Wert des *MsgId*-Feldes im MQMD bleibt allerdings bei Rückgabe des Aufrufs ungeändert, und zwar auch dann, wenn MQMI_NONE oder MQPMO_NEW_MSG_ID angegeben wurden. Wenn die Anwendung die vom Warteschlangenmanager erstellten Nachrichten-IDs kennen muss, muss die Anwendung MQPMR-Datensätze bereitstellen, die das *MsgId*-Feld enthalten.

Die sendende Anwendung kann auch einen anderen Wert als MQMI_NONE für die Nachrichten-ID angeben; der Warteschlangenmanager wird dadurch davon abgehalten, eine eindeutige Nachrichten-ID zu generieren. Eine Anwendung, die eine Nachricht weiterleitet, kann dies nutzen, um die Nachrichten-ID der ursprünglichen Nachricht zu replizieren.

Der Warteschlangenmanager verwendet das Feld für nichts anderes außer um:

- Wie oben beschrieben bei Anforderung einen eindeutigen Wert zu generieren
- Um den Wert an die Anwendung zu liefern, die die Abrufanforderung für die Nachricht ausgibt
- Den Wert des *CorrelId*-Felds jeder Berichtsnachricht zu kopieren, die er bezüglich dieser Nachricht generiert (in Abhängigkeit von den *Report*-Optionen)

Wenn der Warteschlangenmanager oder wenn ein Nachrichtenkanalagent eine Berichtsnachricht generiert, setzt er das *MsgId*-Feld so wie vom *Report*-Feld der ursprünglichen Nachricht angegeben entweder auf MQRO_NEW_MSG_ID oder MQRO_PASS_MSG_ID. Anwendungen, die Berichtsnachrichten erstellen, müssen dies ebenfalls tun.

Beim MQGET-Aufruf ist *MsgId* eines der fünf Felder, mit denen eine bestimmte Nachricht von der Warteschlange abgerufen werden kann. Normalerweise gibt der MQGET-Aufruf die nächste Nachricht in der Warteschlange zurück. Es ist aber auch möglich, eine bestimmte Nachricht anzufordern, indem mindestens eines der fünf Auswahlkriterien in beliebiger Kombination angegeben wird. Es handelt sich um die folgenden Felder:

---

[2] Eine vom Warteschlangenmanager generierte *MsgId* besteht aus einer aus 4 Byte bestehenden Produkt-ID (AMQ ¬ oder CSQ ¬ in ASCII oder EBCDIC, wobei ¬ für ein Leerzeichen steht), gefolgt von einer produktspezifischen Implementierung einer eindeutigen Zeichenfolge. Bei WebSphere MQ enthält sie die ersten 12 Zeichen des Warteschlangenmanagernamens und einen Wert, der von der Systemuhr abgeleitet wurde. Damit sichergestellt wird, dass die Nachrichten-IDs eindeutig sind, müssen die Namen sämtlicher miteinander kommunizierender Warteschlangenmanager sich in den ersten 12 Zeichen unterscheiden. Die Fähigkeit, eine eindeutige Zeichenfolge zu generieren, ist außerdem davon abhängig, dass die Systemuhr nicht zurückgedreht wird. Um die Möglichkeit auszuschließen, dass eine Nachrichten-ID generiert wird, indem der Warteschlangenmanager eine Nachrichten-ID dupliziert, die von der Anwendung generiert wurde, darf die Anwendung keine IDs generieren, deren Anfangszeichen im Bereich A bis I in ASCII oder EBCDIC liegen (X'41' bis X'49' und X'C1' bis X'C9'). Allerdings wird die Anwendung nicht davon abgehalten, IDs mit Anfangszeichen in diesen Bereichen zu erstellen.

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

Die Anwendung setzt mindestens eines der Felder auf den erforderlichen Wert und setzt danach die entsprechenden MQMO_*-Abgleichoptionen im *MatchOptions*-Feld in der MQGMO, um diese Felder als Auswahlkriterien zu verwenden. Nur Nachrichten, die in diesen Feldern über die angegebenen Werte verfügen, sind Kandidaten für einen Abruf. Standardmäßig werden für das *MatchOptions*-Feld (solange keine Änderung durch die Anwendung vorliegt) die Nachrichten-ID und die Korrelations-ID aufeinander abgestimmt.

Bei z/OS sind die verwendbaren Auswahlkriterien durch den Indextyp der Warteschlange beschränkt. In den Informationen zum Warteschlangenattribut *IndexType* finden Sie weitere Details hierzu.

Normalerweise ist die zurückgegebene Nachricht die *erste* Nachricht in der Warteschlange, die den Auswahlkriterien entspricht. Aber wenn MQGMO_BROWSE_NEXT angegeben wird, handelt es sich bei der zurückgegebenen Nachricht um die *nächste* Nachricht, die den Auswahlkriterien entspricht; die Suche nach dieser Nachricht fängt bei der Nachricht an, die auf die aktuelle Cursorposition *folgt*.

Anmerkung: Die Warteschlange wird sequenziell nach einer Nachricht durchsucht, die den Auswahlkriterien entspricht, dementsprechend sind die Abrufzeiten länger, als wenn keine Auswahlkriterien angegeben werden, besonders, wenn viele Nachrichten überprüft werden müssen, bis eine passende gefunden wird. Es gibt die folgenden Ausnahmen:

- einen MQGET-Aufruf von *CorrelId* auf verteilten Plattformen (64 Bit), bei denen dank des *CorrelId*-Indexes keine Notwendigkeit besteht, eine richtige sequenzielle Suche durchzuführen.
- einen MQGET-Aufruf von *IndexType* bei z/OS.

In beiden Fällen wird die Abfrageleistung verbessert.

Weitere Informationen dazu, wie die Auswahlkriterien in verschiedenen Situationen verwendet werden, finden Sie in Tabelle 506 auf Seite 369.

Wird MQMI_NONE als Nachrichten-ID angegeben, hat das dieselbe Wirkung, als würde MQMO_MATCH_MSG_ID *nicht* angegeben, das heißt, dass *jede* Nachrichten-ID passt.

Dieses Feld wird ignoriert, falls beim MQGET-Aufruf im *GetMsgOpts*-Parameter die Option MQGMO_MSG_UNDER_CURSOR angegeben wird.

Bei Rückgabe des MQGET-Aufrufs wird das *MsgId*-Feld auf die Nachrichten-ID der zurückgegebenen Nachricht gesetzt (falls vorhanden).

Folgende Sonderwerte sind zulässig:

MQMI_NONE
Keine Nachrichten-ID angegeben

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQMI_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQMI_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Hierbei handelt es sich um ein Ein-/Ausgabefeld für MQGET-, MQPUT- und MQPUT1-Aufrufe. Die Länge dieses Felds wird durch MQ_MSG_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQMI_NONE.

MsgSeqNumber (MQLONG)
Dies ist die Folgenummer einer logischen Nachricht innerhalb einer Gruppe.

Die Folgenummern beginnen bei 1 und werden für jede neue logische Nachricht in der Gruppe um 1 erhöht. Der maximal zulässige Wert liegt bei 999 999 999. Eine physische Nachricht, die keiner Gruppe angehört, erhält die Folgenummer 1.

In folgenden Fällen muss die Anwendung dieses Feld beim MQPUT- oder MQGET-Aufruf nicht setzen:

- Beim MQPUT-Aufruf ist MQPMO_LOGICAL_ORDER angegeben.
- Beim MQGET-Aufruf ist MQMO_MATCH_MSG_SEQ_NUMBER *nicht* angegeben.

Dies sind die empfohlenen Vorgehensweisen für die Verwendung dieser Aufrufe bei Nachrichten, die keine Berichtsnachrichten sind. Wenn jedoch die Anwendung weitergehend gesteuert werden muss oder es sich bei dem Aufruf um einen MQPUT1-Aufruf handelt, muss die Anwendung sicherstellen, dass *MsgSeqNumber* auf einen geeigneten Wert gesetzt wurde.

Bei Eingabe mit dem MQPUT-oder dem MQPUT1-Aufruf verwendet der Warteschlangenmanager den Wert, der in Physische Reihenfolge in einer Warteschlange beschrieben wird. Bei Ausgabe mit dem MQPUT- oder dem MQPUT1-Aufruf setzt der Warteschlangenmanager das Feld auf den Wert, der mit der Nachricht gesendet wurde.

Bei Eingabe mit dem MQGET-Aufruf verwendet der Warteschlangenmanager den in Tabelle 506 auf Seite 369 angegebenen Wert. Bei Ausgabe des MQGET-Aufrufs setzt der Warteschlangenmanager dieses Feld auf den Wert für die abgerufene Nachricht.

Der Anfangswert dieses Felds ist 1. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQMD_VERSION_2 ist.

MsgType (MQLONG)

Dieses Feld gibt den Typ der Nachricht an. Nachrichtentypen sind folgendermaßen unterteilt:

MQMT_SYSTEM_FIRST
Niedrigster Wert für systemdefinierte Nachrichtentypen.

MQMT_SYSTEM_LAST
Höchster Wert für systemdefinierte Nachrichtentypen.

Folgende Werte sind derzeit im Systembereich definiert:

MQMT_DATAGRAM
Für die Nachricht ist keine Antwort erforderlich.

MQMT_REQUEST
Für die Nachricht ist eine Antwort erforderlich.

Geben Sie den Namen der Warteschlangen an, an die die Antwort im *ReplyToQ*-Feld gesendet werden soll. Das *Report*-Feld gibt an, wie *MsgId* und *CorrelId* der Antwort gesetzt werden müssen.

MQMT_REPLY
Die Nachricht ist die Antwort auf eine frühere Anforderungsnachricht (MQMT_REQUEST). Die Nachricht muss an die Warteschlange gesendet werden, die vom *ReplyToQ*-Feld der Anforderungsnachricht angegeben wurde. Verwenden Sie das *Report*-Feld, um anzugeben, wie *MsgId* und *CorrelId* der Antwort gesetzt werden müssen.

Anmerkung: Der Warteschlangenmanager erzwingt keine Anforderung/Antwort-Beziehung. Dies unterliegt der Zuständigkeit der Anwendung.

MQMT_REPORT
Die Nachricht meldet ein erwartetes oder nicht erwartetes Vorkommnis, das normalerweise mit einer anderen Nachricht in Zusammenhang steht (beispielsweise, wenn eine Anforderungsnachricht erhalten wurde, die nicht gültige Daten enthielt). Senden Sie die Nachricht an die vom *ReplyToQ*-Feld des Nachrichtendeskriptors der ursprünglichen Nachricht angegebene Warteschlange. Setzen Sie das *Feedback*-Feld derart, dass es die Art des Berichts angibt. Verwenden Sie das *Report*-Feld der ursprünglichen Nachricht, um anzugeben, wie *MsgId* und *CorrelId* der Berichtsnachricht gesetzt werden müssen.

Berichtsnachrichten, die vom Warteschlangenmanager oder vom Nachrichtenkanalagenten erstellt werden, werden immer zur *ReplyToQ*-Warteschlange gesendet, wobei das *Feedback*- und das *CorrelId*-Feld wie oben angegeben gesetzt werden.

Es können außerdem von der Anwendung definierte Werte verwendet werden. Sie müssen innerhalb des folgenden Bereichs liegen:

MQMT_APPL_FIRST
Niedrigster Wert für anwendungsdefinierte Nachrichtentypen.

MQMT_APPL_LAST
Höchster Wert für anwendungsdefinierte Nachrichtentypen.

Bei den MQPUT- und MQPUT1-Aufrufen muss der *MsgType*-Wert entweder innerhalb des system- oder des anwendungsdefinierten Bereichs liegen. Ist dies nicht der Fall, schlägt der Aufruf mit dem Ursachencode MQRC_MSG_TYPE_ERROR fehl.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQMT_DATAGRAM.

Offset (MQLONG)

Dies ist die relative Position in Bytes der Daten in der physischen Nachricht ab dem Beginn der logischen Nachricht, der die Daten angehören. Diese Daten werden *Segment* genannt. Die relative Position liegt im Bereich von 0 bis 999.999.999. Eine physische Nachricht, die kein Segment einer logischen Nachricht ist, hat eine relative Position von null.

In folgenden Fällen muss die Anwendung dieses Feld beim MQPUT- oder MQGET-Aufruf nicht setzen:

- Beim MQPUT-Aufruf ist MQPMO_LOGICAL_ORDER angegeben.
- Beim MQGET-Aufruf ist MQMO_MATCH_OFFSET *nicht* angegeben.

Dies sind die empfohlenen Vorgehensweisen für die Verwendung dieser Aufrufe bei Nachrichten, die keine Berichtsnachrichten sind. Wenn jedoch die Anwendung diese Bedingungen nicht erfüllt oder es sich bei dem Aufruf um einen MQPUT1-Aufruf handelt, muss die Anwendung sicherstellen, dass *Offset* auf einen geeigneten Wert gesetzt wurde.

Bei Eingabe mit dem MQPUT-oder dem MQPUT1-Aufruf verwendet der Warteschlangenmanager den Wert, der in Physische Reihenfolge in einer Warteschlange beschrieben wird. Bei Ausgabe mit dem MQPUT- oder dem MQPUT1-Aufruf setzt der Warteschlangenmanager das Feld auf den Wert, der mit der Nachricht gesendet wurde.

Für eine Berichtsnachricht, die ein Segment einer logischen Nachricht meldet, wird - vorausgesetzt, es ist nicht MQOL_UNDEFINED - das Feld *OriginalLength* verwendet, um den Offset in der vom Warteschlangenmanager beibehaltenen Segmentinformation zu aktualisieren.

Bei Eingabe mit dem MQGET-Aufruf verwendet der Warteschlangenmanager den in Tabelle 506 auf Seite 369 angegebenen Wert. Bei Ausgabe des MQGET-Aufrufs setzt der Warteschlangenmanager dieses Feld auf den Wert für die abgerufene Nachricht.

Der Anfangswert dieses Felds ist null. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQMD_VERSION_2 ist.

OriginalLength (MQLONG)

Dieses Feld ist nur relevant für Berichtsnachrichten, die Segmente sind. Es legt die Länge des Nachrichtensegments fest, auf das sich die Berichtsnachricht bezieht; es legt nicht die Länge der logischen Nachricht, zu dem das Segment gehört, oder die Länge der Daten in der Berichtsnachricht fest.

Anmerkung: Beim Erstellen einer Berichtsnachricht für eine Nachricht, die ein Segment ist, kopieren der Warteschlangenmanager und der Nachrichtenkanalagent für die Berichtsnachricht die *GroupId*-, *MsgSeqNumber*-, *Offset*- und *MsgFlags*-Felder der ursprünglichen Nachricht in den MQMD. Dadurch ist die Berichtsnachricht auch ein Segment. Anwendungen, die Berichtsnachrichten erstellen, müssen genauso vorgehen und das *OriginalLength*-Feld ordnungsgemäß setzen.

Der folgende spezielle Wert ist definiert:

MQOL_UNDEFINED
Die ursprüngliche Länge der Nachricht ist nicht definiert.

OriginalLength ist ein Eingabefeld der MQPUT- und MQPUT1-Aufrufe. Der der Wert, den die Anwendung bereitstellt, wird jedoch nur unter bestimmten Umständen akzeptiert:

- Wenn die Nachricht, die eingereiht wird, sowohl ein Segment, als auch eine Berichtsnachricht ist, akzeptiert der Warteschlangenmanager den angegebenen Wert. Folgende Werte sind möglich:
  - Größer als null, wenn das Segment nicht das letzte Segment ist
  - Nicht kleiner als null, wenn das Segment das letzte Segment ist
  - Nicht kleiner als die Länge der in der Nachricht vorhandenen Daten

  Werden diese Bedingungen nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_ORIGINAL_LENGTH_ERROR fehl.
- Wenn die Nachricht, die eingereiht wird, keine Berichtsnachricht, sondern ein Segment ist, ignoriert der Warteschlangenmanager das Feld und verwendet stattdessen die Länge der Anwendungsnachrichtendaten.
- In allen anderen Fällen ignoriert der Warteschlangenmanager das Feld und verwendet stattdessen den Wert MQOL_UNDEFINED.

Dies ist ein Ausgabefeld für den MQGET-Aufruf.

Der Anfangswert dieses Felds ist MQOL_UNDEFINED. Dieses Feld wird ignoriert, wenn die *Version* kleiner ist als MQMD_VERSION_2.

Persistence (MQLONG)

Diese zeigt an, ob die Nachricht bei Systemausfällen und Neustarts des Warteschlangenmanagers beibehalten wird. Bei MQPUT- und MQPUT1-Aufrufen muss der Wert einer der folgenden Möglichkeiten entsprechen:

MQPER_PERSISTENT
Die Nachricht wird bei Systemausfällen und Neustarts des Warteschlangenmanagers beibehalten. Sobald die Nachricht eingereiht und - falls die Nachricht als Teil einer Arbeitseinheit empfangen wird - die Arbeitseinheit, in die sie eingereiht wurde, festgeschrieben wurde, wird die Nachricht im Zusatzspeicher beibehalten. Sie verbleibt dort, bis die Nachricht aus der Warteschlange entfernt und die Arbeitseinheit - falls die Nachricht als Teil einer Arbeitseinheit empfangen wurde - festgeschrieben wurde.

Wenn eine persistente Nachricht an eine ferne Warteschlange gesendet wird, wird sie auf dem Weg zum Empfänger von einem Store-and-forward-Verfahren bei jedem Warteschlangenmanager so lange beibehalten, bis sicher ist, dass sie beim nächsten Warteschlangenmanager eingetroffen ist.

Persistente Nachrichten können in folgenden Warteschlangen nicht platziert werden:

- Temporäre dynamische Warteschlangen
- Gemeinsam genutzte Warteschlangen, die einem CFSTRUCT-Objekt auf CFLEVEL(2) oder darunter zugeordnet sind, oder bei denen das CFSTRUCT-Objekt als RECOVER(NO) definiert ist.

Persistente Nachrichten können in permanente dynamische Warteschlangen und in vordefinierte Warteschlangen eingefügt werden.

MQPER_NOT_PERSISTENT
Die Nachricht wird üblicherweise bei Systemausfällen und Neustarts des Warteschlangenmanagers nicht beibehalten. Dies findet auch dann Anwendung, wenn bei Neustart des Warteschlangenmanagers eine unbeschädigte Kopie der Nachricht im Zusatzspeicher zu finden ist.

Im Fall von NPMCLASS (HIGH)-Warteschlangen werden nicht persistente Nachrichten bei normalem Beenden und Neustarten des Warteschlangenmanagers beibehalten.

Im Fall gemeinsam genutzter Warteschlangen werden nicht persistente Nachrichten bei Neustarts des Warteschlangenmanagers in der Gruppe der gemeinsam genutzten Warteschlangen beibehalten;

allerdings werden sie wiederum bei Ausfällen der Coupling-Facility, die bei gemeinsam genutzten Warteschlangen zum Speichern von Nachrichten verwendet werden, nicht beibehalten.

MQPER_PERSISTENCE_AS_Q_DEF

- Wenn die Warteschlange eine Clusterwarteschlange ist, wird die Persistenz der Nachricht dem *DefPersistence*-Attribut entnommen, das beim *Ziel*-Warteschlangenmanager definiert ist, der der Eigner dieser speziellen Instanz der Warteschlange ist, in die die Nachricht eingereiht wird. Normalerweise haben alle Instanzen einer Clusterwarteschlange denselben Wert für das *DefPersistence*-Attribut, obwohl das nicht vorausgesetzt wird.

  Der Wert von *DefPersistence* wird in das *Persistence*-Feld kopiert, wenn die Nachricht in die Zielwarteschlange eingereiht wird. Sollte *DefPersistence* anschließend geändert werden, hat dies keine Auswirkungen auf die Nachrichten, die bereits in die Warteschlange eingereiht wurden.
- Wenn es sich bei der Warteschlange nicht um eine Clusterwarteschlange handelt, ergibt sich die Persistenz der Nachricht aus dem beim *lokalen* Warteschlangenmanager definierten *DefPersistence*-Attribut, und zwar auch dann, wenn es sich um einen fernen Zielwarteschlangenmanager handelt.

  Gibt es mehr als eine Definition im Auflösungspfad des Warteschlangennamens, wird die Standardpersistenz dem Wert dieses Attributs in der *ersten* Definition im Pfad entnommen. Dieser kann Folgendes einschließen:

  – Eine Aliaswarteschlange
  – Eine lokale Warteschlange
  – Eine lokale Definition einer fernen Warteschlange
  – Ein Warteschlangenmanageralias
  – Eine Übertragungswarteschlange (z. B. die Warteschlange *DefXmitQName*)

  Der Wert von *DefPersistence* wird beim Einreihen der Nachricht in das *Persistence*-Feld kopiert. Wenn *DefPersistence* daraufhin geändert wird, wirkt sich diese Änderung nicht auf Nachrichten aus, die bereits eingereiht wurden.

Sowohl persistente als auch nicht persistente Nachrichten können in derselben Warteschlange vorhanden sein.

Wird eine Nachricht beantwortet, müssen Anwendungen bei der Antwortnachricht die Persistenz der Anforderungsnachricht verwenden.

Bei einem MQGET-Aufruf wird entweder der Wert MQPER_PERSISTENT oder der Wert MQPER_NOT_PERSISTENT zurückgegeben.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQPER_PERSISTENCE_AS_Q_DEF.

Priority (MQLONG)

Bei den MQPUT- und MQPUT1-Aufrufen muss der Wert größer-gleich null sein. Null ist die niedrigste Priorität. Es kann auch der folgende Sonderwert verwendet werden:

MQPRI_PRIORITY_AS_Q_DEF

- Wenn die Warteschlange eine Clusterwarteschlange ist, wird die Priorität der Nachricht dem *DefPriority*-Attribut entnommen, dass beim *Ziel*-Warteschlangenmanager definiert ist, der der Eigner dieser speziellen Instanz der Warteschlange ist, in die die Nachricht eingereiht wird. Normalerweise haben alle Instanzen einer Clusterwarteschlange denselben Wert für das *DefPriority*-Attribut, obwohl das nicht vorausgesetzt wird.

  Der Wert von *DefPriority* wird in das *Priority*-Feld kopiert, wenn die Nachricht in die Zielwarteschlange eingereiht wird. Wenn *DefPriority* daraufhin geändert wird, wirkt sich diese Änderung nicht auf Nachrichten aus, die bereits in die Warteschlange eingereiht wurden.

- Wenn die Warteschlange keine Clusterwarteschlange ist, basiert die Priorität der Nachricht auf dem Attribut *DefPriority*, das im *lokalen* Warteschlangenmanager definiert ist, auch wenn die Zielwarteschlange fern ist.

  Gibt es mehr als eine Definition im Auflösungspfad des Warteschlangennamens, wird die Standardpriorität dem Wert dieses Attributs in der *ersten* Definition im Pfad entnommen. Dieser kann Folgendes einschließen:

  – Eine Aliaswarteschlange
  – Eine lokale Warteschlange
  – Eine lokale Definition einer fernen Warteschlange
  – Ein Warteschlangenmanageralias
  – Eine Übertragungswarteschlange (z. B. die Warteschlange *DefXmitQName*)

  Der Wert von *DefPriority* wird beim Einreihen der Nachricht in das *Priority*-Feld kopiert. Wenn *DefPriority* daraufhin geändert wird, wirkt sich diese Änderung nicht auf Nachrichten aus, die bereits eingereiht wurden.

Der vom MQGET-Aufruf zurückgegebene Wert ist immer größer oder gleich null. Der Wert MQPRI_PRIORITY_AS_Q_DEF wird niemals zurückgegeben.

Wenn eine Nachricht mit einer Priorität eingereiht wird, die den Höchstwert überschreitet, der vom lokalen Warteschlangenmanager unterstützt wird (dieser Höchstwert wird vom *MaxPriority*-Attribut des Warteschlangenmanagers angegeben), wird die Nachricht vom Warteschlangenmanager akzeptiert und mit der höchstmöglichen Priorität des Warteschlangenmanagers von ihm in die Warteschlange eingereiht. Die MQPUT- und MQPUT1-Aufrufe schließen mit MQCC_WARNING und dem Ursachencode MQRC_PRIORITY_EXCEEDS_MAXIMUM ab. Das Feld *Priority* behält allerdings den Wert bei, den die Anwendung angegeben hat, die die Nachricht eingereiht hat.

Wenn bei z/OS eine Nachricht mit der MsgSeqNumber 1 in eine Warteschlange mit einer Reihenfolge bei der Nachrichtenübertragung von MQMDS_PRIORITY und dem Indextyp MQIT_GROUP_ID eingereiht wird, verarbeitet die Warteschlange die Nachricht eventuell mit einer anderen Priorität. Wenn die Nachricht mit einer Priorität von 0 oder 1 in die Warteschlange gestellt wurde, wird sie verarbeitet, als ob sie eine Priorität von 2 hat. Dies liegt daran, dass die Reihenfolge der Nachrichten, die auf diesen Typ von Warteschlange gestellt werden, optimiert wird, um eine effiziente Gruppenvollständigkeitsprüfung zu ermöglichen. Weitere Informationen zur Reihenfolge der Nachrichtenübermittlung MQMDS_PRIORITY und den Indextyp MQIT_GROUP_ID finden Sie im Abschnitt über das MsgDeliverySequence-Attribut.

Wird eine Nachricht beantwortet, müssen Anwendungen bei der Antwortnachricht die Priorität der Anforderungsnachricht verwenden. In anderen Situationen wird durch das Angeben von MQPRI_PRIORITY_AS_Q_DEF ermöglicht, dass die Priorität optimiert werden kann, ohne die Anwendung zu ändern.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQPRI_PRIORITY_AS_Q_DEF.

PutApplName (MQCHAR28)
Dies ist der Name der Anwendung, die die Nachricht eingereiht hat. Er ist Teil des *Ursprungskontextes* der Nachricht. Die Inhalte unterscheiden sich je nach Plattform und eventuell nach Release.

Weitere Informationen zum Nachrichtenkontext finden Sie unter „Überblick für MQMD” auf Seite 400 und Nachrichtenkontext .

Das Format von *PutApplName* ist vom Wert des Feldes *PutApplType* abhängig und kann sich je nach Release unterscheiden. Änderungen sind selten, kommen aber vor, wenn sich die Umgebung ändert.

Wenn der Warteschlangenmanager dieses Feld setzt (dies bezieht sich auf alle Optionen außer MQPMO_SET_ALL_CONTEXT), setzt er es auf einen Wert, der von der Umgebung abhängig ist:

- Verwendet der Warteschlangenmanager bei z/OS:
  – Für z/OS-Stapel den aus 8 Zeichen bestehenden Namen der JES JOB-Karte
  – Für TSO die aus sieben Zeichen bestehende Benutzer-ID der TSO

- Für CICS die aus 8 Zeichen bestehende Anwendungs-ID gefolgt von der aus 4 Zeichen bestehenden Transaktions-ID
- Für IMS die aus 8 Zeichen bestehende IMS-System-ID gefolgt von dem aus 8 Zeichen bestehenden PSB-Namen
- Für XCF den aus 8 Zeichen bestehenden XCF-Gruppennamen gefolgt von dem aus 16 Zeichen bestehenden XCF-Mitgliedsnamen
- Für eine vom Warteschlangenmanager erstellte Nachricht die ersten 28 Zeichen des Warteschlangenmanagernamens
- Für die verteilte Steuerung von Warteschlangen ohne CICS den aus 8 Zeichen bestehenden Jobnamen des Kanalinitiators gefolgt von dem aus 8 Zeichen bestehenden Namen des Moduls, das in die Warteschlange für nicht zustellbare Nachrichten einreiht, gefolgt von einer aus 8 Zeichen bestehenden Task-ID

Die Namen sind jeweils nach rechts mit Leerzeichen aufgefüllt. Dies gilt auch für jedes Leerzeichen im Rest des Felds. Mehrere Namen werden nicht durch ein Trennzeichen voneinander getrennt.

- Bei Windows-Systemen verwendet der Warteschlangenmanager:
  - Bei einer CICS-Anwendung den CICS-Transaktionsnamen
  - Für eine Nicht-CICS-Anwendung die 28 Zeichen ganz rechts im vollständig qualifizierten Namen der ausführbaren Datei
- Unter IBM i verwendet der Warteschlangenmanager den vollständig qualifizierten Jobnamen.
- Bei UNIX-Systemen verwendet der Warteschlangenmanager Folgendes:
  - Bei einer CICS-Anwendung den CICS-Transaktionsnamen
  - Bei Anwendungen, bei denen es sich nicht um CICS-Anwendungen handelt, ruft MQ den Namen des Prozesses aus dem Betriebssytem ab. Dieser wird als Programmdateiname ohne vollständigen Pfad zurückgegeben. Anschließend wird dieser Prozessname von MQ wie folgt in das Feld 'MQMD.PutApplName' eingefügt:

    AIX
    Wenn der Name aus maximal 28 Byte besteht, wird der Name eingefügt und nach rechts mit Leerzeichen aufgefüllt.

    Wenn der Name aus mehr als 28 Byte besteht, werden die 28 Byte des Namens eingefügt, die sich ganz links befinden.

    Linux und Solaris
    Wenn der Name aus maximal 15 Byte besteht, wird der Name eingefügt und nach rechts mit Leerzeichen aufgefüllt.

    Wenn der Name aus mehr als 15 Byte besteht, werden die 15 Byte des Namens eingefügt, die sich ganz links befinden, nach rechts wird mit Leerzeichen aufgefüllt.

    HP-UX
    Wenn der Name aus maximal 14 Byte besteht, wird der Name eingefügt und nach rechts mit Leerzeichen aufgefüllt.

    Wenn der Name aus mehr als 14 Byte besteht, werden die 14 Byte des Namens eingefügt, die sich ganz links befinden, nach rechts wird mit Leerzeichen aufgefüllt.

    Wenn Sie zum Beispiel `/opt/mqm/samp/bin/amqsput QNAME QMNAME` ausführen, lautet der Wert für 'PutApplName' `'amqsput '`. In diesem Feld MQCHAR28 befinden sich 21 Leerzeichen zum Auffüllen. Beachten Sie, dass der vollständige Pfad einschließlich `/opt/mqm/samp/bin` nicht in 'PutApplName' eingeschlossen ist.

Bei den MQPUT- und MQPUT1-Aufrufen ist es, falls MQPMO_SET_ALL_CONTEXT im *PutMsgOpts*-Parameter angegeben wird, ein Ein-/Ausgabefeld. Alle Informationen, die in dem Feld auf ein Nullzeichen folgen, werden verworfen. Das Nullzeichen und alle darauf folgenden Zeichen werden vom Warteschlangenmanager in Leerzeichen konvertiert. Wenn MQPMO_SET_ALL_CONTEXT nicht angegeben wird, wird dieses Feld bei der Eingabe ignoriert und ist ein Nur-Ausgabe-Feld.

PutApplType (MQLONG)

Dies ist die Art der Anwendung, die die Nachricht eingereiht hat. Sie ist Teil des **Ursprungskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten „Überblick für MQMD” auf Seite 400 und Nachrichtenkontext.

PutApplType kann einem der folgenden Standardtypen entsprechen. Sie können auch Ihre eigenen Typen definieren, aber nur mit Werten im Bereich von MQAT_USER_FIRST bis MQAT_USER_LAST.

MQAT_AIX
AIX-Anwendung (selber Wert wie MQAT_UNIX)

MQAT_BROKER
Broker.

MQAT_CICS
CICS-Transaktion

MQAT_CICS_BRIDGE
CICS-Brücke.

MQAT_CICS_VSE
CICS/VSE-Transaktion

MQAT_DOS
WebSphere MQ MQI-Clientanwendung auf PC DOS

MQAT_DQM
Verteilter Warteschlangenmanageragent

MQAT_GUARDIAN
Tandem Guardian-Anwendung (gleicher Wert wie MQAT_NSK).

MQAT_IMS
IMS-Anwendung

MQAT_IMS_BRIDGE
IMS-Brücke.

MQAT_JAVA
Java

MQAT_MVS
MVS- oder TSO-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_NOTES_AGENT
Lotus Notes Agent-Anwendung

MQAT_NSK
HP Integrity NonStop Server-Anwendung

MQAT_OS390
OS/390-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_OS400
IBM i-Anwendung

MQAT_QMGR
Warteschlangenmanager

MQAT_UNIX
UNIX-Anwendung

MQAT_VOS
Stratus VOS-Anwendung.

MQAT_WINDOWS
16-Bit-Windows-Anwendung

MQAT_WINDOWS_NT
32-Bit-Windows-Anwendung

MQAT_WLM
z/OS-Workload-Manager-Anwendung

MQAT_XCF
XCF.

MQAT_ZOS
z/OS-Anwendung

MQAT_DEFAULT
Standardanwendungstyp

Dies ist der Standardanwendungstyp für die Plattform, auf der die Anwendung ausgeführt wird.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung. Kompilieren Sie aus diesem Grund die Anwendung immer unter Verwendung des Headers, berücksichtigen oder KOPIEREN Sie Dateien, die für die Plattform, auf der die Anwendung ausgeführt werden wird, geeignet sind.

MQAT_UNKNOWN
Verwenden Sie diesen Wert, um anzugeben, dass der Anwendungstyp unbekannt ist, obwohl andere Kontextinformationen vorhanden sind.

MQAT_USER_FIRST
Niedrigster Wert für einen benutzerdefinierten Anwendungstyp

MQAT_USER_LAST
Höchster Wert für einen benutzerdefinierten Anwendungstyp

Folgender Sonderwert kann auftreten:

MQAT_NO_CONTEXT
Dieser Wert wird vom Warteschlangenmanager gesetzt, wenn eine Nachricht ohne Kontext eingereiht wird (das heißt, wenn die Kontextoption MQPMO_NO_CONTEXT angegeben ist).

Wenn eine Nachricht abgerufen wird, kann *PutApplType* auf diesen Wert überprüft werden, um zu entscheiden, ob die Nachricht über Kontext verfügt (es wird empfohlen, *PutApplType* von einer Anwendung, die MQPMO_SET_ALL_CONTEXT verwendet, nie auf MQAT_NO_CONTEXT setzen zu lassen, falls auch nur eines der anderen Kontextfelder belegt ist).

Wenn der Warteschlangenmanager als Ergebnis des Einreihens durch eine Anwendung diese Informationen generiert, ist der Wert, auf den das Feld gesetzt wird, von der Umgebung abhängig. Bei IBM i wird er auf MQAT_OS400 gesetzt; der Warteschlangenmanager verwendet bei IBM i nie MQAT_CICS.

Bei den MQPUT- und MQPUT1-Aufrufen ist es, falls MQPMO_SET_ALL_CONTEXT im *PutMsgOpts*-Parameter angegeben wird, ein Ein-/Ausgabefeld. Wenn MQPMO_SET_ALL_CONTEXT nicht angegeben wird, wird dieses Feld bei der Eingabe ignoriert und ist ein Nur-Ausgabe-Feld.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Der Anfangswert dieses Felds ist MQAT_NO_CONTEXT.

PutDate (MQCHAR8)

Dies ist das Datum, an dem die Nachricht eingereiht wurde. Es ist Teil des **Ursprungskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten „Überblick für MQMD” auf Seite 400 und Nachrichtenkontext.

Das Format, das für das vom Warteschlangenmanager in diesem Feld generierte Datum verwendet wird, lautet:

- YYYYMMDD,

wobei die Zeichen Folgendes darstellen:

YYYY
Jahr (vier Ziffern)

MM
Monat des Jahres (01 bis 12)

TT
Tag des Monats (01 bis 31)

Für die *PutDate*- und *PutTime*-Felder wird in Abhängigkeit davon, dass sie Systemuhr genau auf GMT gesetzt wird, Greenwich Mean Time (GMT) verwendet.

Wenn die Nachricht als Teil einer Arbeitseinheit eingereiht wurde, entspricht das Datum dem Datum, an dem die Nachricht eingereiht wurde, und nicht dem Datum, an dem die Arbeitseinheit festgeschrieben wurde.

Bei den MQPUT- und MQPUT1-Aufrufen ist es, falls MQPMO_SET_ALL_CONTEXT im *PutMsgOpts*-Parameter angegeben wird, ein Ein-/Ausgabefeld. Der Inhalt des Felds wird vom Warteschlangenmanager nicht geprüft. Nur Informationen, die auf ein Nullzeichen im Feld folgen, werden gelöscht. Der Warteschlangenmanager wandelt das Nullzeichen und alle anschließenden Zeichen in Leerzeichen um. Wenn MQPMO_SET_ALL_CONTEXT nicht angegeben wird, wird dieses Feld bei der Eingabe ignoriert und ist ein Nur-Ausgabe-Feld.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Die Länge dieses Felds wird durch MQ_PUT_DATE_LENGTH angegeben. Der Anfangswert dieses Felds ist in der Programmiersprache C eine Nullzeichenfolge und acht Leerzeichen in anderen Programmiersprachen.

PutTime (MQCHAR8)

Dies ist die Zeit, zu der die Nachricht eingereiht wurde. Sie ist Teil des **Ursprungskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten „Überblick für MQMD" auf Seite 400 und Nachrichtenkontext.

Das Format, das für die vom Warteschlangenmanager in diesem Feld generierte Uhrzeit verwendet wird, lautet:

- HHMMSSTH,

wobei die Zeichen Folgendes repräsentieren (Angaben entsprechen der Reihenfolge):

HH
Stunde (00 bis 23)

MM
Minute (00 bis 59)

SS
Sekunden (00 bis 59; siehe Anmerkung)

T
Zehntelsekunden (0 bis 9)

H
Hundertstelsekunden (0 bis 9)

Anmerkung: Wenn die Systemuhr mit einem sehr präzisen Zeitnormal synchronisiert ist, kann es in seltenen Fällen vorkommen, dass für die Sekunden 60 oder 61 in *PutTime* zurückgegeben wird. Dies geschieht, wenn Schaltsekunden in den globalen Zeitstandard aufgenommen werden.

Für die *PutDate*- und *PutTime*-Felder wird in Abhängigkeit davon, dass sie Systemuhr genau auf GMT gesetzt wird, Greenwich Mean Time (GMT) verwendet.

Wenn die Nachricht als Teil einer Arbeitseinheit eingereiht wurde, entspricht die Zeit dem Zeitpunkt, an dem die Nachricht eingereiht wurde, und nicht dem Zeitpunkt, an dem die Arbeitseinheit festgeschrieben wurde.

Bei den MQPUT- und MQPUT1-Aufrufen ist es, falls MQPMO_SET_ALL_CONTEXT im *PutMsgOpts*-Parameter angegeben wird, ein Ein-/Ausgabefeld. Abgesehen davon, dass jede Information, die innerhalb des Felds auf ein Nullzeichen folgt, gelöscht wird, überprüft der Warteschlangenmanager den Inhalt des Felds nicht. Der Warteschlangenmanager wandelt das Nullzeichen und alle anschließenden Zeichen in Leerzeichen um. Wenn MQPMO_SET_ALL_CONTEXT nicht angegeben wird, wird dieses Feld bei der Eingabe ignoriert und ist ein Nur-Ausgabe-Feld.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Die Länge dieses Felds wird durch MQ_PUT_TIME_LENGTH angegeben. Der Anfangswert dieses Felds ist in der Programmiersprache C eine Nullzeichenfolge und acht Leerzeichen in anderen Programmiersprachen.

ReplyToQ (MQCHAR48)

Gibt den Namen der Nachrichtenwarteschlange an, an die die Anwendung, von der die Abrufanforderung (GET) für die Nachricht abgesetzt wurde, MQMT_REPLY- und MQMT_REPORT-Nachrichten sendet. Der Name gibt den lokalen Namen der Warteschlange an, so wie er auf dem mit *ReplyToQMgr* angegebenen Warteschlangenmanager definiert ist. Diese Warteschlange darf keine Modellwarteschlange sein, obwohl der sendende Warteschlangenmanager dies beim Einreihen der Nachricht nicht überprüft.

Für die MQPUT- und MQPUT1-Aufrufe darf dieses Feld nicht leer sein, falls das Feld *MsgType* den Wert MQMT_REQUEST hat oder falls vom Feld *Report* Berichtsnachrichten angefordert werden. Der angegebene (oder ersetzte) Wert wird jedoch unabhängig vom Nachrichtentyp an die Anwendung weitergegeben, die die Abrufanforderung für die Nachricht ausgibt.

Wenn das Feld *ReplyToQMgr* leer ist, überprüft der lokale Warteschlangenmanager den *ReplyToQ*-Namen in seinen eigenen Warteschlangendefinitionen. Wenn eine lokale Definition einer fernen Warteschlange mit diesem Namen existiert, wird der *ReplyToQ*-Wert in der übertragenen Nachricht durch den Wert des *RemoteQName*-Attributs der Definition der fernen Warteschlange ersetzt, und dieser Wert wird im Nachrichtendeskriptor zurückgegeben, wenn die empfangende Anwendung einen MQGET-Aufruf für die Nachricht absetzt. Liegt keine lokale Definition einer fernen Warteschlange vor, bleibt *ReplyToQ* unverändert.

Wenn der Name angegeben wird, kann er mit abschließenden Leerzeichen aufgefüllt sein; das erste Nullzeichen und nachfolgende Zeichen werden wie Leerzeichen behandelt. Sonst wird nicht überprüft, ob der Name die Namensregeln für Warteschlangen erfüllt; dies trifft auch auf den übertragenen Namen zu, falls *ReplyToQ* in der übertragenen Nachricht ersetzt wird. Es wird lediglich geprüft, dass ein Name angegeben wurde, wenn dies erforderlich ist.

Wenn keine Empfangswarteschlange für Antworten erforderlich ist, setzen Sie das Feld *ReplyToQ* auf Leerzeichen oder (in der Programmiersprache C) auf die Nullzeichenfolge oder auf mindestens ein Leerzeichen, gefolgt von einem Nullzeichen; das Feld muss initialisiert sein.

Für den MQGET-Aufruf gibt der Warteschlangenmanager immer den Namen mit Leerzeichen aufgefüllt bis zur Länge des Felds zurück.

Wenn eine Nachricht, die eine Berichtsnachricht erfordert, nicht bereitgestellt und auch die Berichtsnachricht nicht an die angegebene Warteschlange übermittelt werden kann, werden sowohl die ursprüngliche Nachricht als auch die Berichtsnachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht (weitere Informationen finden Sie in der Beschreibung des Attributs *DeadLetterQName* im Abschnitt „Attribute für den Warteschlangenmanager" auf Seite 797).

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ReplyToQMgr (MQCHAR48)

Gibt den Namen des Warteschlangenmanagers an, an den die Antwortnachricht oder Berichtsnachricht gesendet werden soll. *ReplyToQ* ist der lokale Name einer Warteschlange, die für den lokalen Warteschlangenmanager definiert ist.

Wenn das Feld *ReplyToQMgr* leer ist, überprüft der lokale Warteschlangenmanager den *ReplyToQ*-Namen in seinen Warteschlangendefinitionen. Wenn eine lokale Definition einer fernen Warteschlange mit diesem Namen existiert, wird der *ReplyToQMgr*-Wert in der übertragenen Nachricht durch den Wert des Attributs *RemoteQMgrName* der Definition der fernen Warteschlange ersetzt und dieser Wert wird im Nachrichtendeskriptor zurückgegeben, wenn die empfangende Anwendung einen MQGET-Aufruf für die Nachricht absetzt. Liegt eine lokale Definition einer fernen Warteschlange nicht vor, handelt es sich bei dem mit der Nachricht übertragenen *ReplyToQMgr* um den Namen des lokalen Warteschlagenmanagers.

Wenn der Name angegeben wird, kann er mit abschließenden Leerzeichen aufgefüllt sein; das erste Nullzeichen und nachfolgende Zeichen werden wie Leerzeichen behandelt. Sonst wird nicht überprüft, ob der Name die Namensregeln für Warteschlangenmanager erfüllt; dies trifft auch auf den übertragenen Namen zu, falls *ReplyToQMgr* in der übertragenen Nachricht ersetzt wird.

Wenn keine Empfangswarteschlange für Antworten erforderlich ist, setzen Sie das Feld *ReplyToQMgr* auf Leerzeichen oder (in der Programmiersprache C) auf die Nullzeichenfolge oder auf mindestens ein Leerzeichen, gefolgt von einem Nullzeichen; das Feld muss initialisiert sein.

Für den MQGET-Aufruf gibt der Warteschlangenmanager immer den Namen mit Leerzeichen aufgefüllt bis zur Länge des Felds zurück.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

Report (MQLONG)

Bei einer Berichtsnachricht handelt es sich um eine Nachricht zu einer andere Nachricht, die eine Anwendung über erwartete bzw. unerwartete Ereignisse bei der ursprünglichen Nachricht unterrichtet. Das *Report*-Feld ermöglicht der Anwendung, die die ursprüngliche Nachricht sendet, anzugeben, welche Berichtsnachrichten erforderlich sind, ob die Anwendungsnachrichtendaten enthalten sein müssen, und außerdem, wie bei Berichts- und Antwortnachrichten die Nachrichten- und Korrelations-IDs gesetzt werden sollen. Jede oder alle (oder keine) der folgenden Arten von Berichtsnachrichten können angefordert werden:

- Ausnahmebedingung
- Ablauf
- Bestätigung bei Eingang (COA)
- Bestätigung bei Zustellung (COD)
- Benachrichtigung über positive Aktion (PAN)
- Benachrichtigung über negative Aktion (NAN)

Wenn mehr als eine Art von Berichtsnachricht oder andere Berichtsoptionen erforderlich sind, können die Werte:

- gemeinsam hinzugefügt werden (dieselbe Konstante nicht mehr als einmal hinzufügen) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

Die Anwendung, die die Berichtsnachricht empfängt, kann den Grund für die Erstellung des Berichts durch eine Prüfung des Feldes *Feedback* im MQMD bestimmen; weitere Einzelheiten finden Sie im Feld *Feedback*.

Die Verwendung von Berichtsoptionen beim Einreihen einer Nachricht zu einem Thema kann zur Erstellung von keiner oder einer Berichtsnachricht oder auch vielen Berichtsnachrichten führen, die an die Anwendung gesendet werden. Dies liegt daran, dass die Veröffentlichungsnachricht an keine oder eine subskribierende Anwendung oder auch an viele subskribierende Anwendungen gesendet werden kann.

Ausnahmeoptionen: Geben Sie eine der angegebenen Optionen an, um eine Ausnahmeberichtsnachricht anzufordern.

MQRO_EXCEPTION
: Ein Nachrichtenkanalagent erstellt diese Art von Bericht, wenn eine Nachricht an einen anderen Warteschlangenmanager gesendet wird und die Nachricht nicht an die angegebene Zielwarteschlange übermittelt werden kann. Beispielsweise können die Zielwarteschlange oder eine temporäre Übertragungswarteschlange voll sein oder die Nachricht kann zu groß für die Warteschlange sein.

  Ob ein Ausnahmeberichtsnachricht erstellt wird, ist abhängig von der Persistenz der ursprünglichen Nachricht und von der Geschwindigkeit des Nachrichtenkanals (normal oder schnell), den die ursprüngliche Nachricht durchläuft:

- Bei allen persistenten Nachrichten und bei nicht persistenten Nachrichten, die normale Nachrichtenkanäle durchlaufen, wird der Ausnahmebericht *nur* erstellt, wenn die durch die sendende Anwendung für die Fehlerbedingung angegebene Aktion erfolgreich abgeschlossen werden kann. Die sendende Anwendung kann eine der folgenden Aktionen angeben, um bei Vorliegen der Fehlerbedingung die Disposition der ursprünglichen Nachricht zu steuern:
  - MQRO_DEAD_LETTER_Q (die ursprüngliche Nachricht wird der Warteschlange für nicht zustellbare Nachrichten zugeordnet).
  - MQRO_DISCARD_MSG (die ursprüngliche Nachricht wird gelöscht).

  Wenn die von der sendenden Anwendung angegebene Aktion nicht erfolgreich abgeschlossen werden kann, verbleibt die ursprüngliche Nachricht in der Übertragungswarteschlange und es wird keine Ausnahmeberichtsnachricht erstellt.
- Bei nicht persistenten Nachrichten, die schnelle Nachrichtenkanäle durchlaufen, wird die ursprüngliche Nachricht aus der Übertragungswarteschlange entfernt und der Ausnahmebericht erstellt, *auch wenn* die für die Fehlerbedingung angegebene Aktion nicht erfolgreich abgeschlossen werden kann. Wenn zum Beispiel MQRO_DEAD_LETTER_Q angegeben wird, die ursprüngliche Nachricht aber nicht in der Warteschlange für nicht zustellbare Nachrichten eingereiht werden kann, weil die Warteschlange voll ist, wird der Ausnahmebericht erstellt und die ursprüngliche Nachricht wird gelöscht.

  Weitere Informationen zu normalen und schnellen Nachrichtenkanälen finden Sie unter Nicht persistente Nachrichtengeschwindigkeit (NPMSPEED) .

Es wird kein Ausnahmebericht erstellt, wenn die Anwendung, die die ursprüngliche Nachricht eingereiht hat, synchron durch den Ursachencode über das Problem benachrichtigt werden kann, der von dem MQPUT- oder MQPUT1-Aufruf zurückgegeben wird.

Anwendungen können außerdem Ausnahmeberichte senden, um anzuzeigen, dass eine Nachricht nicht verarbeitet werden kann (beispielsweise, wenn es sich um eine Lastschrift handelt, die dazu führen würde, dass das Kreditlimit des Kontos überschritten werden würde).

Nachrichtendaten der ursprünglichen Nachricht sind nicht Teil der Berichtsnachricht.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA und MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA
Diese Ausnahme entspricht MQRO_EXCEPTION, wobei allerdings die ersten 100 Bytes der Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden. Falls die ursprüngliche Nachricht mindestens eine MQ-Headerstruktur enthält, wird diese in der Berichtsnachricht zusätzlich zu den 100 Bytes Anwendungsdaten mit angegeben.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA und MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA
Ausnahmebericht mit vollständigen Daten erforderlich

Dies entspricht MQRO_EXCEPTION, mit der Ausnahme, dass alle Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA und MQRO_EXCEPTION_WITH_FULL_DATA.

Ablaufoptionen: Geben Sie eine der angegebenen Optionen an, um eine Ablaufberichtsnachricht anzufordern.

MQRO_EXPIRATION
Der Warteschlangenmanager erstellt diese Art von Bericht, wenn die Nachricht vor der Übermittlung an eine Anwendung gelöscht wird, weil die Ablaufzeit abgelaufen ist (siehe das Feld *Expiry*). Wenn diese Option nicht gesetzt wird, wird keine Berichtsnachricht erstellt, falls aus diesem Grund eine Nachricht gelöscht wird (auch dann nicht, wenn Sie eine der MQRO_EXCEPTION_*-Optionen angeben).

Nachrichtendaten der ursprünglichen Nachricht sind nicht Teil der Berichtsnachricht.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA und MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA
Diese Ausnahme entspricht MQRO_EXPIRATION, wobei allerdings die ersten 100 Bytes der Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden. Falls die ursprüngliche Nachricht mindestens eine MQ-Headerstruktur enthält, wird diese in der Berichtsnachricht zusätzlich zu den 100 Bytes Anwendungsdaten mit angegeben.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA und MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA
Dies entspricht MQRO_EXPIRATION, mit der Ausnahme, dass alle Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA und MQRO_EXPIRATION_WITH_FULL_DATA.

Optionen für Bestätigung bei Eingang (COA): Geben Sie eine der angegebenen Optionen an, um einen Bericht mit Bestätigung bei Eingang anzufordern.

MQRO_COA
Diese Art von Bericht wird von dem Warteschlangenmanager generiert, der zu dem Zeitpunkt, an dem die Nachricht in die Zielwarteschlange eingereiht wird, Eigner der Zielwarteschlange ist. Nachrichtendaten der ursprünglichen Nachricht sind nicht Teil der Berichtsnachricht.

Wenn die Nachricht als Teil einer Arbeitseinheit eingereiht wird und die Zielwarteschlange eine lokale Warteschlange ist, kann der Bericht mit Bestätigung bei Eingang, der von dem Warteschlangenmanager generiert wird, nur dann abgerufen werden, wenn die Arbeitseinheit festgeschrieben wird.

Es wird kein Bericht mit Bestätigung bei Eingang generiert, wenn das Feld *Format* im Nachrichtendeskriptor MQFMT_XMIT_Q_HEADER oder MQFMT_DEAD_LETTER_HEADER ist. Dadurch wird verhindert, dass ein Bericht mit Bestätigung bei Eingang generiert wird, wenn die Nachricht in eine Übertragungswarteschlange eingereiht oder unzustellbar ist und in eine Warteschlange für nicht zustellbare Nachrichten eingereiht wird.

Wenn eine IMS-Brückenwarteschlange vorliegt, wird ein Bericht mit Bestätigung bei Eingang generiert, wenn die Nachricht die IMS-Warteschlange erreicht (Bestätigung von IMS erhalten) und nicht, wenn die Nachricht in die MQ-Brückenwarteschlange eingereiht wird. Das bedeutet, dass, falls IMS nicht aktiv ist, kein Bericht mit Bestätigung bei Eingang generiert wird, bevor nicht IMS gestartet und eine Nachricht in der IMS-Warteschlange eingereiht wurde.

Der Benutzer, der ein Programm ausführt, das eine Nachricht mit MQMD.Report=MQRO_COA einreiht, muss über die Berechtigung '+passid' für die Antwortwarteschlange verfügen. Wenn der Benutzer nicht über die Berechtigung '+passid' verfügt, erreicht die COA-Berichtsnachricht die Antwortwarteschlange nicht. Es wird versucht, die Berichtsnachricht in die Warteschlange für nicht zustellbare Nachrichten zu stellen.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COA, MQRO_COA_WITH_DATA und MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_DATA
Diese Ausnahme entspricht MQRO_COA, wobei allerdings die ersten 100 Bytes der Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden. Falls die ursprüngliche Nachricht mindestens eine MQ-Headerstruktur enthält, wird diese in der Berichtsnachricht zusätzlich zu den 100 Bytes Anwendungsdaten mit angegeben.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COA, MQRO_COA_WITH_DATA und MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_FULL_DATA
Dies entspricht MQRO_COA, mit der Ausnahme, dass alle Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COA, MQRO_COA_WITH_DATA und MQRO_COA_WITH_FULL_DATA.

Optionen für Bestätigung bei Zustellung (COD): Geben Sie eine der angegebenen Optionen an, um eine Bericht mit Bestätigung bei Zustellung anzufordern.

MQRO_COD
Diese Art von Bericht wird von dem Warteschlangenmanager generiert, wenn eine Anwendung die Nachricht von der Warteschlange so abruft, dass die Nachricht aus der Warteschlange gelöscht wird. Nachrichtendaten der ursprünglichen Nachricht sind nicht Teil der Berichtsnachricht.

Wenn die Nachricht als Teil einer Arbeitseinheit abgerufen wird, wird die Berichtsnachricht innerhalb derselben Arbeitseinheit generiert, sodass der Bericht erst dann verfügbar ist, wenn die Arbeitseinheit festgeschrieben wird. Wenn die Arbeitseinheit zurückgesetzt wird, wird der Bericht nicht gesendet.

Wenn eine Nachricht mit der Option MQGMO_MARK_SKIP_BACKOUT abgerufen wird, wird nicht immer ein Bericht mit Bestätigung bei Zustellung generiert. Wenn die primäre Arbeitseinheit zurückgesetzt, aber die sekundäre Arbeitseinheit festgeschrieben wird, wird eine Nachricht aus der Warteschlange entfernt, aber kein Bericht mit Bestätigung bei Zustellung generiert.

Es wird kein Bericht mit Bestätigung bei Zustellung generiert, wenn das Feld *Format* im Nachrichtendeskriptor MQFMT_DEAD_LETTER_HEADER ist. Dadurch wird verhindert, dass ein Bericht mit Bestätigung bei Zustellung generiert wird, falls die Nachricht unzustellbar sein sollte und in eine Warteschlange für nicht zustellbare Nachrichten eingereiht wird.

MQRO_COD ist nicht gültig, falls die Zielwarteschlange eine XCF-Warteschlange ist.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COD, MQRO_COD_WITH_DATA und MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_DATA
Diese Ausnahme entspricht MQRO_COD, wobei allerdings die ersten 100 Bytes der Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden. Falls die ursprüngliche Nachricht mindestens eine MQ-Headerstruktur enthält, wird diese in der Berichtsnachricht zusätzlich zu den 100 Bytes Anwendungsdaten mit angegeben.

Wenn bei dem MQGET-Aufruf der ursprünglichen Nachricht MQGMO_ACCEPT_TRUNCATED_MSG angegeben wird und die abgerufene Nachricht abgeschnitten ist, hängt die Menge der Anwendungsnachrichtendaten, die Teil der Berichtsnachricht sind, von der Umgebung ab:

- Bei z/OS handelt es sich mindestens um:
  - die Länge der ursprünglichen Nachricht
  - die Länge des Puffers, der beim Abrufen der Nachricht verwendet wird,
  - 100 Bytes.
- Bei anderen Umgebungen handelt es sich mindestens um:
  - die Länge der ursprünglichen Nachricht
  - 100 Bytes.

MQRO_COD_WITH_DATA ist nicht gültig, falls die Zielwarteschlange eine XCF-Warteschlange ist.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COD, MQRO_COD_WITH_DATA und MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_FULL_DATA
Dies entspricht MQRO_COD, mit der Ausnahme, dass alle Anwendungsnachrichtendaten der ursprünglichen Nachricht in der Berichtsnachricht mit angegeben werden.

MQRO_COD_WITH_FULL_DATA ist nicht gültig, falls die Zielwarteschlange eine XCF-Warteschlange ist.

Geben Sie bei folgenden Ausnahmen jeweils nicht mehr als eine an: MQRO_COD, MQRO_COD_WITH_DATA und MQRO_COD_WITH_FULL_DATA.

Optionen für Benachrichtigungen über Aktionen: Geben Sie eine oder beide der gelisteten Optionen an, um anzufordern, dass die empfangende Anwendung eine Berichtsnachricht über eine positive oder über eine negative Nachricht sendet.

MQRO_PAN
Diese Art von Bericht wird von der Anwendung generiert, die die Nachricht abruft und auf sie reagiert. Er zeigt an, dass die in der Nachricht angeforderte Aktion erfolgreich ausgeführt wurde. Die Anwendung, die den Bericht generiert, bestimmt, ob im Bericht Daten eingeschlossen werden sollen.

Abgesehen davon, dass er diese Anforderung an die Anwendung übermittelt, die diese Nachricht abruft, führt der Warteschlangenmanager basierend auf dieser Option keine Aktionen aus. Die abrufende Anwendung muss, falls angebracht, den Bericht generieren.

MQRO_NAN
Diese Art von Bericht wird von der Anwendung generiert, die die Nachricht abruft und auf sie reagiert. Er zeigt an, dass die in der Nachricht angeforderte Aktion *nicht* erfolgreich ausgeführt wurde. Die Anwendung, die den Bericht generiert, bestimmt, ob im Bericht Daten eingeschlossen werden sollen. Sie können beispielsweise einige Daten einschließen, die angeben, aus welchem Grund die Anforderung nicht ausgeführt werden konnte.

Abgesehen davon, dass er diese Anforderung an die Anwendung übermittelt, die diese Nachricht abruft, führt der Warteschlangenmanager basierend auf dieser Option keine Aktionen aus. Die abrufende Anwendung muss, falls angebracht, den Bericht generieren.

Die Anwendung muss bestimmen, welche Bedingungen einer positiven Aktion und welche einer negativen Aktion entsprechen. Wenn die Anforderung allerdings nur zum Teil ausgeführt wurde, generieren Sie bei Anforderung besser einen NAN- statt eines PAN-Berichts. Jede mögliche Bedingung muss entweder einer positiven oder einer negativen Aktion entsprechen, jedoch nicht beiden Arten von Aktionen.

Optionen der Nachrichten-ID: Geben Sie eine der beiden gelisteten Optionen an, um zu bestimmen, wie die *MsgId* der Berichtsnachricht (oder der Antwortnachricht) gesetzt werden muss.

MQRO_NEW_MSG_ID
Dies ist die Standardaktion und gibt an, dass, wenn aufgrund dieser Nachricht ein Bericht oder eine Antwort generiert wird, für den Bericht oder die Antwortnachricht eine neue *MsgId* generiert wird.

MQRO_PASS_MSG_ID
Wenn aufgrund dieser Nachricht ein Bericht oder eine Antwort generiert wird, wird die *MsgId* dieser Nachricht in die *MsgId* des Berichts oder der Antwortnachricht kopiert.

Die *MsgId* ist für jeden Subskribenten, der eine Kopie der Veröffentlichung erhält, unterschiedlich und aus diesem Grund ist die *MsgId*, die in den Bericht oder die Antwortnachricht kopiert wird, jedes Mal verschieden.

Wenn diese Option nicht angegeben ist, wird MQRO_NEW_MSG_ID vorausgesetzt.

Optionen der Korrelations-ID: Geben Sie eine der aufgeführten Optionen an, um festzulegen, wie die *CorrelId* der Berichtsnachricht (oder der Antwortnachricht) gesetzt werden soll.

MQRO_COPY_MSG_ID_TO_CORREL_ID
Dies ist die Standardaktion, die angibt, dass, wenn aufgrund dieser Nachricht ein Bericht oder eine Antwort generiert wird, die *MsgId* dieser Nachricht in die *CorrelId* des Berichts oder der Antwortnachricht kopiert wird.

Die *MsgId* einer Veröffentlichungsnachricht ist für jeden Subskribenten unterschiedlich, der eine Kopie der Veröffentlichung erhält, sodass die *MsgId*, die in die *CorrelId* der Berichts- oder Antwortnachricht kopiert wird, für jede Kopie unterschiedlich ist.

MQRO_PASS_CORREL_ID
Wenn aufgrund dieser Nachricht ein Bericht oder eine Antwort generiert wird, wird die *CorrelId* dieser Nachricht in die *CorrelId* des Berichts oder der Antwortnachricht kopiert.

Die *CorrelId* einer Veröffentlichungsnachricht ist spezifisch für einen Subskribenten, es sei denn, sie verwendet die Option MQSO_SET_CORREL_ID und setzt das Feld SubCorrelId im MQSD auf MQCI_NONE. Deswegen ist es möglich, dass die *CorrelId*, die in die *CorrelId* des Berichts oder der Antwortnachricht kopiert wird, für jeden einzelnen Fall unterschiedlich ist.

Wenn diese Option nicht angegeben ist, wird MQRO_COPY_MSG_ID_TO_CORREL_ID vorausgesetzt.

Server, die Anforderungen beantworten oder Berichtsnachrichten erstellen, müssen prüfen, ob in der ursprünglichen Nachricht die MQRO_PASS_MSG_ID- oder MQRO_PASS_CORREL_ID-Optionen gesetzt waren. Sollte das der Fall sein, müssen die Server die Aktionen ausführen, die für diese Optionen angegeben sind. Ist keine der Optionen gesetzt, müssen die Server die entsprechende Standardaktion ausführen.

Dispositionsoptionen: Geben Sie eine der aufgeführten Optionen an, um die Disposition der ursprünglichen Nachricht zu steuern, wenn sie nicht der Zielwarteschlange übermittelt werden kann. Die Anwendung kann die Dispositionsoptionen unabhängig von der Anforderung von Abweichungsberichten setzen.

MQRO_DEAD_LETTER_Q
Dies ist die Standardaktion, die die Nachricht in die Warteschlange für nicht zustellbare Nachrichten einreiht, wenn die Nachricht nicht in die Zielwarteschlange übermittelt werden kann. Dies geschieht in den folgenden Situationen:

- Die Anwendung, die die ursprüngliche Nachricht eingereiht hat, kann nicht gleichzeitig mit dem Ursachencode, der vom MQPUT- oder MQPUT1-Aufruf zurückgegeben wurde, über das Problem informiert werden. Ein Abweichungsbericht wird generiert, falls ein solcher vom Sender angefordert wurde.
- Die Anwendung, die die ursprüngliche Nachricht eingereiht hat, hat sie zu einem Thema eingereiht.

MQRO_DISCARD_MSG
Diese Option löscht die Nachricht, falls sie nicht der Zielwarteschlange übermittelt werden kann. Dies geschieht in den folgenden Situationen:

- Die Anwendung, die die ursprüngliche Nachricht eingereiht hat, kann nicht gleichzeitig mit dem Ursachencode, der vom MQPUT- oder MQPUT1-Aufruf zurückgegeben wurde, über das Problem informiert werden. Ein Abweichungsbericht wird generiert, falls ein solcher vom Sender angefordert wurde.
- Die Anwendung, die die ursprüngliche Nachricht eingereiht hat, hat sie zu einem Thema eingereiht.

Wenn Sie die ursprüngliche Nachricht an den Sender zurückgeben wollen, ohne dass die ursprüngliche Nachricht in der Warteschlange für nicht zustellbare Nachrichten eingereiht wird, muss der Sender MQRO_DISCARD_MSG mit MQRO_EXCEPTION_WITH_FULL_DATA angeben.

MQRO_PASS_DISCARD_AND_EXPIRY
Wenn diese Option für eine Nachricht gesetzt wird und aus diesem Grund wird ein Bericht oder eine Antwort generiert, erbt der Nachrichtendeskriptor des Berichts Folgendes:

- MQRO_DISCARD_MSG, falls diese Option gesetzt wurde.
- Die verbleibende Ablaufzeit der Nachricht, wenn es sich nicht um einen Ablaufbericht handelt. Sollte es ein Ablaufbericht sein, wird die Ablaufzeit auf 60 Sekunden gesetzt.

Aktivitätsoption

MQRO_ACTIVITY
Wird dieser Wert verwendet, kann die Route **jeder** Nachricht durch ein Warteschlangenmanagernetz aufgezeichnet werden. Die Berichtsoption kann für jede aktuelle Benutzernachricht angegeben werden, sodass Sie in kürzester Zeit die Route der Nachricht durch das Netz berechnen können.

Falls die Anwendung, die die Nachricht generiert, die Aktivitätenberichte nicht aktivieren kann, können die Berichte mithilfe von von Warteschlangenmanagern bereitgestellten API-Cross-Exits aktiviert werde.

Anmerkung:

1. Je weniger Warteschlangenmanager im Netz in der Lage sind, Aktivitätenberichte zu generieren, desto weniger ausführlich ist die Route.

2. Die Aktivitätenberichte sind eventuell nur schwer in die richtige Reihenfolge zu bringen, um die Route zu bestimmen.
3. Die Aktivitätenberichte finden eventuell keine Route zum angeforderten Bestimmungsort.
4. Nachrichten mit dieser Berichtsoption müssen von jedem Warteschlangenmanager akzeptiert werden, auch wenn er die Option nicht versteht. Dies ermöglicht, dass die Berichtsoption auf jede Benutzernachricht gesetzt werden kann, selbst dann, wenn sie von einem Warteschlangenmanager verarbeitet wird, der nicht Version 6.0 oder höher ist.
5. Wenn entweder ein Warteschlangenmanagerprozess oder ein Benutzerprozess eine Aktivität für eine Nachricht mit dieser Optionsgruppe ausführt, kann er einen Aktivitätenbericht generieren und einreihen.

Standardoption: Geben Sie das Folgende an, wenn keine Berichtsoptionen erforderlich sind:

MQRO_NONE
Verwenden Sie diesen Wert, um anzugeben, dass keine anderen Optionen angegeben wurden. MQRO_NONE dient zur Unterstützung der Programmdokumentation. Diese Option ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann die Verwendung nicht erkannt werden.

Allgemeine Informationen:

1. Alle erforderlichen Berichtstypen müssen explizit von der Anwendung angefordert werden, die die ursprüngliche Nachricht gesendet hat. Wenn beispielsweise ein COA-Bericht angefordert wird, jedoch kein Ausnahmebericht, wird ein COA-Bericht erstellt, wenn die Nachricht in die Zielwarteschlange eingereiht wird, jedoch kein Ausnahmebericht, wenn die Zielwarteschlange voll ist, wenn die Nachricht dort eintrifft. Wenn keine *Report*-Optionen festgelegt werden, werden vom Warteschlangenmanager und vom Nachrichtenkanalagenten (MCA) keine Berichtsnachrichten generiert.

   Einige Berichtsoptionen können angegeben werden, auch wenn sie vom lokalen Warteschlangenmanager nicht erkannt werden. Dies ist hilfreich, wenn die Option vom *Ziel*-Warteschlangenmanager verarbeitet wird. Weitere Informationen finden Sie in „Report options and message flags” auf Seite 904.

   Wenn eine Berichtsnachricht angefordert wird, muss der Name der Warteschlange, an die der Bericht gesendet werden soll, im Feld *ReplyToQ* angegeben werden. Wenn eine Berichtsnachricht empfangen wird, kann die Spezifik des Berichts durch Überprüfung des Felds *Feedback* im Nachrichtendeskriptor bestimmt werden.
2. Wenn der Warteschlangenmanager oder der Nachrichtenkanalagent, der die Berichtsnachricht erstellt, die Berichtsnachricht nicht in die Antwortwarteschlange einreihen kann, weil beispielsweise die Antwortwarteschlange oder die Übertragungswarteschlange voll ist, wird die Berichtsnachricht stattdessen in die Warteschlange für nicht zustellbare Nachrichten eingereiht. Wenn auch dies *fehlschlägt* oder keine Warteschlange für nicht zustellbare Nachrichten existiert, hängt es von der Art der Berichtsnachricht ab, welche Aktion ausgewählt wird:
   - Wenn es sich bei der Berichtsnachricht um einen Abweichungsbericht handelt, verbleibt die Nachricht, die den Abweichungsbericht generiert hat, in der Übertragungswarteschlange. Damit wird sichergestellt, dass die Nachricht nicht verloren geht.
   - Bei allen anderen Berichtstypen wird die Berichtsnachricht gelöscht und die Verarbeitung normal fortgesetzt. Dies liegt daran, dass die ursprüngliche Nachricht entweder bereits sicher geliefert wurde (COA- oder COD-Berichtsnachrichten) oder nicht mehr von Interesse ist (Ablaufberichtsnachricht).

   Wenn eine Berichtsnachricht erfolgreich in eine Warteschlange eingereiht wurde (entweder die Zielwarteschlange oder eine temporäre Übertragungswarteschlange), unterliegt sie keiner speziellen Verarbeitung mehr, sondern wird wie jede andere Nachricht behandelt.
3. Wenn der Bericht generiert wird, wird unter Verwendung der Berechtigung von *UserIdentifier* im MQMD der Nachricht, die den Bericht ausgelöst hat, die Warteschlange *ReplyToQ* geöffnet und die Berichtsnachricht eingereiht. Ausnahmen bilden die folgenden Fälle:

- Ausnahmeberichte, die von einem empfangenden MCA erstellt werden, werden mit einer beliebigen Berechtigung eingereiht, die der MCA bei dem Versuch verwendet hat, die Nachricht einzureihen, die den Bericht verursacht hat.
- Vom Warteschlangenmanager erstellte COA-Berichte, werden mit einer beliebigen Berechtigung eingereiht, die verwendet wurde, als die Nachricht, die diesen Bericht verursacht hat, in den Warteschlangenmanager eingereiht wurde, der den Bericht erstellt hat. Wenn die Nachricht beispielsweise von einem empfangenden MCA mit der Benutzer-ID des MCA eingereiht wurde, reiht der Warteschlangenmanager den COA-Bericht mit der Benutzer-ID des MCA ein.

Anwendungen, die Berichte erstellen, müssen dieselbe Berechtigung verwenden, die sie zur Generierung einer Antwort verwenden; normalerweise handelt es sich dabei um die Berechtigung der Benutzer-ID in der ursprünglichen Nachricht.

Wenn der Bericht an ein fernes Ziel übermittelt werden muss, können Sender und Empfänger auf dieselbe Weise wie bei anderen Nachrichten entscheiden, ob sie ihn akzeptieren.

4. Bei Anforderung einer Berichtsnachricht mit Daten:
   - Die Berichtsnachricht wird immer mit dem vom Sender der ursprünglichen Nachricht angeforderten Datenvolumen erstellt. Wenn die Berichtsnachricht zu groß für die Antwortwarteschlange ist, wird sie auf die oben beschriebene Weise verarbeitet; die Berichtsnachricht wird nie abgeschnitten, um sie in die Antwortwarteschlange einzufügen.
   - Wenn das *Format* der ursprünglichen Nachricht MQFMT_XMIT_Q_HEADER ist, schließen die im Bericht berücksichtigten Daten MQXQH nicht mit ein. Die Berichtsdaten beginnen mit dem ersten Byte der Daten jenseits des MQXQH in der ursprünglichen Nachricht. Dies geschieht unabhängig davon, ob es sich bei der Warteschlange um eine Übertragungswarteschlange handelt.
5. Wenn eine COA-, COD- oder Ablaufberichtsnachricht von der Antwortwarteschlange empfangen wird, wird garantiert, dass die ursprüngliche Nachricht eingegangen ist, geliefert wurde bzw. abgelaufen ist. Jedoch kann, wenn mindestens eine dieser Berichtsnachrichten angefordert und *nicht* empfangen wurde, nicht vom Gegenteil ausgegangen werden, denn eine der folgenden Möglichkeiten könnte eingetreten sein:
   a. Die Berichtsnachricht ist wegen eines inaktiven Links verzögert.
   b. Die Berichtsnachricht wurde blockiert, da an der temporären Übertragungswarteschlange oder Antwortwarteschlange eine Blockbedingung existiert (die Warteschlange ist beispielsweise voll oder für Einreihungen gesperrt).
   c. Die Berichtsnachricht ist in einer Warteschlange für nicht zustellbare Nachrichten eingereiht.
   d. Als der Warteschlangenmanager versuchte, die Berichtsnachricht zu generieren, konnte er sie weder in die richtige Warteschlange, noch in die Warteschlange für nicht zustellbare Nachrichten einreihen. Aus diesem Grund konnte die Berichtsnachricht nicht generiert werden.
   e. Zwischen der Meldung der Aktion (Eingang, Bereitstellung oder Ablauf) und der Generierung der entsprechenden Berichtsnachricht ist ein Fehler des Warteschlangenmanager aufgetreten. (Dies tritt nicht bei COD-Berichtsnachrichten auf, wenn die Anwendung die ursprüngliche Nachricht in einer Arbeitseinheit abruft, da die COD-Berichtsnachricht in derselben Arbeitseinheit erstellt wird.)

   Abweichungsberichte können aufgrund der oben erwähnten Möglichkeiten 1, 2 und 3 auf dieselbe Weise verzögert werden. Wenn aber ein Nachrichtenkanalagent keine Ausnahmeberichtsnachricht generieren kann (die Berichtsnachricht kann weder in die Antwortwarteschlange noch in die Warteschlange für nicht zustellbare Nachrichten eingereiht werden), verbleibt die ursprüngliche Nachricht beim Sender in der Übertragungswarteschlange und der Kanal wird geschlossen. Dies tritt unabhängig davon auf, ob die Berichtsnachricht am sendenden oder empfangenden Ende des Kanals erstellt wurde.
6. Falls die ursprüngliche Nachricht temporär geblockt ist (woraufhin eine Ausnahmeberichtsnachricht generiert und die ursprüngliche Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wird), die Blockierung dann aber beseitigt ist und eine Anwendung daraufhin die ursprüngliche Nachricht von der Warteschlange für nicht zustellbare Nachrichten liest und sie erneut an ihren Bestimmungsort einreiht, kann das Folgende auftreten:

- Obwohl eine Ausnahmeberichtsnachricht erstellt wurde, wird die ursprüngliche Nachricht letztendlich an ihrem Ziel empfangen.
- In Bezug auf eine einzelne ursprüngliche Nachricht wird mehr als eine Ausnahmeberichtsnachricht generiert, denn die ursprüngliche Nachricht wird eventuell später erneut blockiert.

Berichtsnachrichten beim Einreihen in ein Themas:

1. Wenn eine Nachricht in ein Thema eingereiht wird, können Berichte generiert werden. Diese Nachricht wird an alle Subskribenten des Themas gesendet. Dabei kann es sich um keinen, einen oder viele Subskribenten handeln. Dies sollte bei der Entscheidung für die Verwendung der Berichtsoptionen berücksichtigt werden, da möglicherweise viele Berichtsnachrichten erstellt werden.
2. Beim Einreihen einer Nachricht in ein Thema können viele Zielwarteschlangen vorhanden sein, denen eine Kopie der Nachricht hinzugefügt werden muss. Wenn bei einigen der Zielwarteschlangen ein Problem wie eine volle Warteschlange auftritt, ist der erfolgreiche Abschluss des Aufrufs MQPUT von der Einstellung von NPMSGDLV oder PMSGDLV abhängig (je nach Persistenz der Nachricht). Wenn die Einstellung besagt, dass die Lieferung der Nachricht an die Zielwarteschlange erfolgreich abgeschlossen werden muss (beispielsweise bei einer persistenten Nachricht an einen permanenten Subskribenten, bei der PMSGDLV auf ALL oder ALLDUR eingestellt ist), ist der Erfolg durch Erfüllung einer der folgenden Bedingungen definiert:
   - Erfolgreiches Einreihen in der Warteschlange für Teilnehmerberechtigungen
   - Verwenden von MQRO_DEAD_LETTER_Q und ein erfolgreiches Einreihen in der Warteschlange für nicht zustellbare Nachrichten, wenn die Warteschlange für Teilnehmerberechtigungen die Nachricht nicht abrufen kann.
   - Verwenden von MQRO_DISCARD_MSG, falls die Warteschlange für Teilnehmerberechtigungen die Nachricht nicht abrufen kann.

Berichtsnachrichten für Nachrichtensegmente:

1. Berichtsnachrichten können für Nachricht angefordert werden, die Segmentierung zulassen (weitere Informationen finden Sie in der Beschreibung des Flags MQMF_SEGMENTATION_ALLOWED). Falls der Warteschlangenmanager die Nachricht segmentieren muss, kann für jedes der Segmente, das anschließend der relevanten Bedingung entspricht, eine Berichtsnachricht generiert werden. Anwendungen müssen darauf vorbereitet werden, für jede Art der angeforderten Berichtsnachrichten mehrere Berichtsnachrichten zu erhalten. Verwenden Sie das Feld *GroupId* der Berichtsnachricht, um die mehrfachen Berichte mit der Gruppen-ID der ursprünglichen Nachricht zu korrelieren, und das Feld *Feedback*, um den Typ jeder Berichtsnachricht anzugeben.
2. Wenn MQGMO_LOGICAL_ORDER verwendet wird, um Berichtsnachrichten für Segmente abzurufen, seien Sie sich bewusst, dass die anschließenden MQGET-Aufrufe *verschiedene Arten* von Berichten zurückgeben können. Wenn zum Beispiel sowohl COA- als auch COD-Berichte für eine Nachricht, die vom Warteschlangenmanager segmentiert wird, angefordert werden, können die MQGET-Aufrufe der Berichtsnachrichten die COA- und COD-Berichte auf eine unvorhersehbare Weise verzahnt zurückgeben. Vermeiden Sie das, indem Sie die MQGMO_COMPLETE_MSG-Option verwenden (optional mit MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG sorgt dafür, dass der Warteschlangenmanager Berichtsnachrichten neu erstellt, die demselben Berichtstyp entsprechen. Beispielsweise könnte der erste MQGET-Aufruf alle COA-Nachrichten neu erstellen, die sich auf die ursprüngliche Nachricht beziehen, und der zweite MQGET-Aufruf könnte alle COD-Nachrichten neu erstellen. Welche zuerst neu erstellt werden, hängt davon ab, welche Art von Berichtsnachricht zuerst in der Warteschlange auftritt.
3. Anwendungen, die selbst Segmente einreihen, können für jedes Segment unterschiedliche Berichtsoptionen angeben. Beachten Sie jedoch die folgenden Aspekte:
   - Falls die Segmente mit der Option MQGMO_COMPLETE_MSG abgerufen werden, werden nur die Berichtsoptionen im *ersten* Segment vom Warteschlangenmanager berücksichtigt.
   - Wenn die Segmente eines nach dem anderen abgerufen werden und die meisten über eine der MQRO_COD_*-Optionen verfügen, aber mindestens ein Segment nicht darüber verfügt, können Sie weder die MQGMO_COMPLETE_MSG-Option verwenden, um die Berichtsnachrichten mit einem

einzigen MQGET-Aufruf abzurufen, noch die MQGMO_ALL_SEGMENTS_AVAILABLE-Option, um zu erkennen, ob alle Berichtsnachrichten eingetroffen sind.

4. In einem MQ-Netz können die Warteschlangenmanager über unterschiedliche Leistungsmerkmale verfügen. Wenn eine Berichtsnachricht für ein Segment von einem Warteschlangenmanager oder Nachrichtenkanalagenten generiert wird, der Segmentierung nicht unterstützt, bezieht der Warteschlangenmanager bzw. der Nachrichtenkanalagent nicht standardmäßig die erforderlichen Segmentinformationen in die Berichtsnachricht mit ein. Dadurch kann es schwierig werden, die ursprüngliche Nachricht zu ermitteln, aufgrund derer der Bericht generiert wurde. Vermeiden Sie dieses Problem, indem Sie mit der Berichtsnachricht Daten anfordern, das heißt, indem Sie die entsprechenden Optionen von MQRO_*_WITH_DATA oder MQRO_*_WITH_FULL_DATA angeben. Beachten Sie jedoch, dass, falls MQRO_*_WITH_DATA angegeben wird, *weniger als* 100 Bytes an Anwendungsnachrichtendaten an die Anwendung, die die Berichtsnachricht abruft, zurückgegeben werden, falls die Berichtsnachricht von einem Warteschlangenmanager bzw. Nachrichtenkanalagenten generiert wird, der keine Segmentierung unterstützt.

Inhalt des Nachrichtendeskriptors für eine Berichtsnachricht: Wenn der Warteschlangenmanager oder Nachrichtenkanalagent (MCA) eine Berichtsnachricht erstellt, setzt er die Felder im Nachrichtendeskriptor auf die folgenden Werte und reiht die Nachricht dann wie gewöhnlich ein.

Feld im MQMD	**Verwendeter Wert**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	MQRO_NONE
MsgType	MQMT_REPORT
Expiry	MQEI_UNLIMITED
Feedback	Entsprechend der Art des Berichts (MQFB_COA, MQFB_COD, MQFB_EXPIRATION oder ein MQRC_*-Wert)
Encoding	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
CodedCharSetId	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
Format	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
Priority	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
Persistence	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
MsgId	Gemäß Berichtsoptionen im ursprünglichen Nachrichtendeskriptor
CorrelId	Gemäß Berichtsoptionen im ursprünglichen Nachrichtendeskriptor
BackoutCount	0
ReplyToQ	Leerzeichen
ReplyToQMgr	Warteschlangenmanagername
UserIdentifier	Wie von der MQPMO_PASS_IDENTITY_CONTEXT-Option angegeben
AccountingToken	Wie von der MQPMO_PASS_IDENTITY_CONTEXT-Option angegeben
ApplIdentityData	Wie von der MQPMO_PASS_IDENTITY_CONTEXT-Option angegeben
PutApplType	MQAT_QMGR oder wie es dem Nachrichtenkanalagenten entspricht
PutApplName	Erste 28 Bytes des Namens des Warteschlangenmanagers oder des Nachrichtenkanalagenten. Bei Berichtsnachrichten, die von der IMS-Brücke erstellt werden, enthält dieses Feld den XCF-Gruppennamen und den XCF-Mitgliedsnamen des IMS-Systems, auf das sich die Nachricht bezieht.
PutDate	Datum, an dem die Berichtsnachricht gesendet wird

Feld im MQMD	Verwendeter Wert
PutTime	Zeit, zu der die Berichtsnachricht gesendet wird
ApplOriginData	Leerzeichen
GroupId	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
MsgSeqNumber	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
Offset	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
MsgFlags	Kopiert aus dem ursprünglichen Nachrichtendeskriptor
OriginalLength	Kopiert aus dem ursprünglichen Nachrichtendeskriptor, falls nicht MQOL_UNDEFINED zutrifft, und sonst auf die Länge der ursprünglichen Nachrichtendaten festgelegt

Eine Anwendung, die Berichte geniert, muss ähnliche Werte festlegen, mit folgenden Ausnahmen:

- Das Feld *ReplyToQMgr* kann auf leer gesetzt werden (der Warteschlangenmanager ändert dies in den Namen des lokalen Warteschlangenmanagers, wenn die Nachricht eingereiht wird).
- Verwenden Sie die Option, die für eine Antwort verwendet worden wäre, normalerweise MQPMO_PASS_IDENTITY_CONTEXT, um die Kontextfelder festzulegen.

Analysieren des Berichtsfeldes: Das Feld *Report* enthält Unterfelder. Aus diesem Grund müssen Anwendungen, die überprüfen müssen, ob der Sender einer Nachricht einen bestimmten Bericht angefordert hat, eines der Verfahren verwenden, die im Abschnitt „Analysieren des Berichtsfelds" auf Seite 906 beschrieben werden.

Dies ist ein Ausgabefeld für den MQGET-Aufruf und ein Eingabefeld für den MQPUT- und den MQPUT1-Aufruf. Der Anfangswert dieses Felds ist MQRO_NONE.

StrucId (MQCHAR4)

Dies ist die Struktur-ID und muss Folgendem entsprechen:

MQMD_STRUC_ID
ID für Nachrichtendeskriptorstruktur.

Für die Programmiersprache C ist auch die Konstante MQMD_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQMD_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQMD_STRUC_ID.

UserIdentifier (MQCHAR12)

Dieses Attribut ist Bestandteil des **Identitätskontextes** der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie unter „Überblick für MQMD" auf Seite 400 und Nachrichtenkontext .

UserIdentifier gibt die Benutzer-ID der Anwendung an, von der die Nachricht stammt. Der Warteschlangenmanager behandelt diese Informationen als Zeichendaten, ohne jedoch ihr Format zu definieren.

Verwenden Sie nach dem Erhalt einer Nachricht *UserIdentifier* im Feld *AlternateUserId* des Parameters *ObjDesc* eines nachfolgenden MQOPEN- oder MQPUT1-Aufrufs, um die Berechtigungsprüfung für den Benutzer *UserIdentifier* auszuführen, statt die Anwendung öffnen zu lassen.

Wenn der Warteschlangenmanager diese Information für einen MQPUT- oder MQPUT1-Aufruf generiert:

- Unter z/OS verwendet der Warteschlangenmanager die *AlternateUserId* des Parameters *ObjDesc* des MQOPEN- oder MQPUT1-Aufrufs, wenn die Option MQOO_ALTERNATE_USER_AUTHORITY oder MQPMO_ALTERNATE_USER_AUTHORITY angegeben wurde. Wenn die relevante Option nicht angegeben wurde, verwendet der Warteschlangenmanager eine von der Umgebung festgelegte Benutzer-ID.

- In anderen Umgebungen verwendet der Warteschlangenmanager immer eine von der Umgebung festgelegte Benutzer-ID.

Wenn die Benutzer-ID durch die Umgebung vorgegeben wird, gilt Folgendes:

- Verwendet der Warteschlangenmanager bei z/OS:
  - Für MVS (Stapel) die Benutzer-ID der Jobkarte des JES oder der gestarteten Task
  - Für TSO die Benutzer-ID, die während der Jobfreigabe an den Job weitergegeben wurde
  - Für CICS die der Task zugeordnete Benutzer-ID
  - Für IMS hängt die Benutzer-ID von der Art der Anwendung ab:
    - Für:
      - BMP-Bereiche ohne Nachrichten
      - IFP-Bereiche ohne Nachrichten
      - BMP- und IFP-Bereiche mit Nachrichten, die *keinen* erfolgreichen GU-Aufruf ausgegeben haben

      verwendet der Warteschlangenmanager die Benutzer-ID der JES JOB-Karte für den Bereich oder die TSO-Benutzer-ID. Wenn diese leer oder gleich null sind, verwendet er den Namen des Programmspezifikationsblocks (PSB).
    - Für:
      - BMP- und IFP-Bereichen mit Nachrichten, die einen erfolgreichen GU-Aufruf ausgegeben *haben*
      - MPP-Bereiche

      verwendet der Warteschlangenmanager eine der folgenden Möglichkeiten:
      - Die der Nachricht zugeordnete angemeldete Benutzer-ID.
      - Der Name des logischen Terminals (LTERM)
      - Die Benutzer-ID der Jobkarte des JES des Bereichs
      - Die Benutzer-ID der TSO
      - Der Name des Programmspezifikationsblocks
- Bei IBM i verwendet der Warteschlangenmanager den Namen des Benutzerprofils, das dem Anwendungsjob zugeordnet ist.
- Bei UNIX-Systemen verwendet der Warteschlangenmanager Folgendes:
  - Den Anmeldenamen der Anwendung
  - Die aktuelle Benutzer-ID des Prozesses, falls kein Anmeldename verfügbar ist
  - Die der Transaktion zugeordnete Benutzer-ID, falls es sich bei der Anwendung um eine CICS-Transaktion handelt
- Bei Windows-Systemen verwendet der Warteschlangenmanager die ersten 12 Zeichen des angemeldeten Benutzernamens.

Bei diesem Feld handelt es sich normalerweise um ein vom Warteschlangenmanager generiertes Ausgabefeld, aber bei einem MQPUT- oder MQPUT1-Aufruf können Sie das Feld als Ein-/Ausgabefeld definieren und das Benutzer-ID-Feld angeben, statt diese Information vom Warteschlangenmanager generieren zu lassen. Geben Sie im Parameter PutMsgOpts entweder MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT und im Feld UserIdentifier eine Benutzer-ID an, wenn der Warteschlangenmanager das Feld UserIdentifier bei einem MQPUT- oder MQPUT1-Aufruf nicht generieren soll.

Bei MQPUT- und MQPUT1-Aufrufen ist dies ein Ein-/Ausgabefeld, falls MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT im Parameter *PutMsgOpts* angegeben werden. Alle Informationen, die in dem Feld auf ein Nullzeichen folgen, werden verworfen. Der Warteschlangenmanager wandelt das Nullzeichen und alle anschließenden Zeichen in Leerzeichen um. Wenn MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT nicht angegeben werden, wird dieses Feld bei der Eingabe ignoriert oder ist ein Nur-Ausgabe-Feld.

Nach erfolgreichem Beenden des MQPUT- oder MQPUT1-Aufrufs enthält dieses Feld den *UserIdentifier*, der mit der Nachricht übertragen wurde, wenn sie in eine Warteschlange eingereiht wurde. Dabei handelt es sich um den Wert von *UserIdentifier*, der mit der Nachricht, falls sie beibehalten wird, aufbewahrt wird (weitere Informationen zu ständigen Veröffentlichungen finden Sie in der Beschreibung von MQPMO_RETAIN), aber nicht als *UserIdentifier* verwendet wird, wenn die Nachricht als Veröffentlichung an Subskribenten gesendet wird, denn sie stellen einen Wert bereit, *UserIdentifier* in allen, an sie gesandten Veröffentlichungen außer Kraft zu setzen. Wenn die Nachricht keinen Kontext aufweist, ist das Feld leer.

Dies ist ein Ausgabefeld für den MQGET-Aufruf. Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 12 leere Zeichen in anderen Programmiersprachen.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Es muss sich um eine der folgenden Möglichkeiten handeln:

MQMD_VERSION_1
Version-1 Nachrichtendeskriptorstruktur.

Diese Option wird in allen Umgebungen unterstützt.

MQMD_VERSION_2
Nachrichtendeskriptorstruktur der Version 2

Diese Version wird in allen Umgebungen mit WebSphere MQ Version 6.0 und höher sowie von WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind, unterstützt.

Anmerkung: Wenn ein MQMD Version-2 verwendet wird, führt der Warteschlangenmanager zusätzliche Prüfungen jeder MQ-Headerstruktur aus, die sich am Anfang der Anwendungsnachrichtendaten befindet. Weitere Details finden Sie in den Hinweisen zur Verwendung des MQPUT-Aufrufs.

Nur in der neueren Version der Struktur verwendete Felder werden in den Feldbeschreibungen entsprechend gekennzeichnet. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQMD_CURRENT_VERSION
Aktuelle Version der Nachrichtendeskriptorstruktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQMD_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQMD

Tabelle 515. Anfangswerte von Feldern in MQMD für MQMD

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQMD_STRUC_ID	'MD'
Version	MQMD_VERSION_1	1
Report	MQRO_NONE	0
MsgType	MQMT_DATAGRAM	8
Expiry	MQEI_UNLIMITED	-1
Feedback	MQFB_NONE	0
Encoding	MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	MQCCSI_Q_MGR	0
Format	MQFMT_NONE	Leerzeichen
Priority	MQPRI_PRIORITY_AS_Q_DEF	-1
Persistence	MQPER_PERSISTENCE_AS_Q_DEF	2

Tabelle 515. Anfangswerte von Feldern in MQMD für MQMD (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
MsgId	MQMI_NONE	Nullen
CorrelId	MQCI_NONE	Nullen
BackoutCount	--	0
ReplyToQ	--	Nullzeichenfolge oder Leerzeichen.
ReplyToQMgr	--	Nullzeichenfolge oder Leerzeichen.
UserIdentifier	--	Nullzeichenfolge oder Leerzeichen.
AccountingToken	MQACT_NONE	Nullen
ApplIdentityData	--	Nullzeichenfolge oder Leerzeichen.
PutApplType	MQAT_NO_CONTEXT	0
PutApplName	--	Nullzeichenfolge oder Leerzeichen.
PutDate	--	Nullzeichenfolge oder Leerzeichen.
PutTime	--	Nullzeichenfolge oder Leerzeichen.
ApplOriginData	--	Nullzeichenfolge oder Leerzeichen.
GroupId	MQGI_NONE	Nullen
MsgSeqNumber	--	1
Offset	--	0
MsgFlags	MQMF_NONE	0
OriginalLength	MQOL_UNDEFINED	-1

Anmerkungen:

1. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
2. In der Programmiersprache C enthält die MakrovariableMQMD_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQMD MyMD = {MQMD_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    Report;            /* Options for report messages */
```

```
  MQLONG    MsgType;           /* Message type */
  MQLONG    Expiry;            /* Message lifetime */
  MQLONG    Feedback;          /* Feedback or reason code */
  MQLONG    Encoding;          /* Numeric encoding of message data */
  MQLONG    CodedCharSetId;    /* Character set identifier of message
                                  data */
  MQCHAR8   Format;            /* Format name of message data */
  MQLONG    Priority;          /* Message priority */
  MQLONG    Persistence;       /* Message persistence */
  MQBYTE24  MsgId;             /* Message identifier */
  MQBYTE24  CorrelId;          /* Correlation identifier */
  MQLONG    BackoutCount;      /* Backout counter */
  MQCHAR48  ReplyToQ;          /* Name of reply queue */
  MQCHAR48  ReplyToQMgr;       /* Name of reply queue manager */
  MQCHAR12  UserIdentifier;    /* User identifier */
  MQBYTE32  AccountingToken;   /* Accounting token */
  MQCHAR32  ApplIdentityData;  /* Application data relating to
                                  identity */
  MQLONG    PutApplType;       /* Type of application that put the
                                  message */
  MQCHAR28  PutApplName;       /* Name of application that put the
                                  message */
  MQCHAR8   PutDate;           /* Date when message was put */
  MQCHAR8   PutTime;           /* Time when message was put */
  MQCHAR4   ApplOriginData;    /* Application data relating to origin */
  MQBYTE24  GroupId;           /* Group identifier */
  MQLONG    MsgSeqNumber;      /* Sequence number of logical message
                                  within group */
  MQLONG    Offset;            /* Offset of data in physical message
                                  from start of logical message */
  MQLONG    MsgFlags;          /* Message flags */
  MQLONG    OriginalLength;    /* Length of original message */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQMD based,
  3 StrucId          char(4),       /* Structure identifier */
  3 Version          fixed bin(31), /* Structure version number */
  3 Report           fixed bin(31), /* Options for report messages */
  3 MsgType          fixed bin(31), /* Message type */
  3 Expiry           fixed bin(31), /* Message lifetime */
  3 Feedback         fixed bin(31), /* Feedback or reason code */
  3 Encoding         fixed bin(31), /* Numeric encoding of message
                                       data */
  3 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                       message data */
  3 Format           char(8),       /* Format name of message data */
  3 Priority         fixed bin(31), /* Message priority */
  3 Persistence      fixed bin(31), /* Message persistence */
  3 MsgId            char(24),      /* Message identifier */
  3 CorrelId         char(24),      /* Correlation identifier */
  3 BackoutCount     fixed bin(31), /* Backout counter */
  3 ReplyToQ         char(48),      /* Name of reply queue */
  3 ReplyToQMgr      char(48),      /* Name of reply queue manager */
  3 UserIdentifier   char(12),      /* User identifier */
  3 AccountingToken  char(32),      /* Accounting token */
  3 ApplIdentityData char(32),      /* Application data relating to
                                       identity */
  3 PutApplType      fixed bin(31), /* Type of application that put the
                                       message */
  3 PutApplName      char(28),      /* Name of application that put the
                                       message */
  3 PutDate          char(8),       /* Date when message was put */
  3 PutTime          char(8),       /* Time when message was put */
  3 ApplOriginData   char(4),       /* Application data relating to
                                       origin */
  3 GroupId          char(24),      /* Group identifier */
  3 MsgSeqNumber     fixed bin(31), /* Sequence number of logical
                                       message within group */
  3 Offset           fixed bin(31), /* Offset of data in physical
                                       message from start of logical
                                       message */
  3 MsgFlags         fixed bin(31), /* Message flags */
  3 OriginalLength   fixed bin(31); /* Length of original message */
```

Deklaration in High Level Assembler

```
MQMD                   DSECT
MQMD_STRUCID           DS   CL4   Structure identifier
MQMD_VERSION           DS   F     Structure version number
MQMD_REPORT            DS   F     Options for report messages
MQMD_MSGTYPE           DS   F     Message type
MQMD_EXPIRY            DS   F     Message lifetime
MQMD_FEEDBACK          DS   F     Feedback or reason code
MQMD_ENCODING          DS   F     Numeric encoding of message data
MQMD_CODEDCHARSETID    DS   F     Character set identifier of message
*                                 data
MQMD_FORMAT            DS   CL8   Format name of message data
MQMD_PRIORITY          DS   F     Message priority
MQMD_PERSISTENCE       DS   F     Message persistence
```

```
MQMD_MSGID              DS   XL24  Message identifier
MQMD_CORRELID           DS   XL24  Correlation identifier
MQMD_BACKOUTCOUNT       DS   F     Backout counter
MQMD_REPLYTOQ           DS   CL48  Name of reply queue
MQMD_REPLYTOQMGR        DS   CL48  Name of reply queue manager
MQMD_USERIDENTIFIER     DS   CL12  User identifier
MQMD_ACCOUNTINGTOKEN    DS   XL32  Accounting token
MQMD_APPLIDENTITYDATA   DS   CL32  Application data relating to identity
MQMD_PUTAPPLTYPE        DS   F     Type of application that put the
*                                  message
MQMD_PUTAPPLNAME        DS   CL28  Name of application that put the
*                                  message
MQMD_PUTDATE            DS   CL8   Date when message was put
MQMD_PUTTIME            DS   CL8   Time when message was put
MQMD_APPLORIGINDATA     DS   CL4   Application data relating to origin
MQMD_GROUPID            DS   XL24  Group identifier
MQMD_MSGSEQNUMBER       DS   F     Sequence number of logical message
*                                  within group
MQMD_OFFSET             DS   F     Offset of data in physical message
*                                  from start of logical message
MQMD_MSGFLAGS           DS   F     Message flags
MQMD_ORIGINALLENGTH     DS   F     Length of original message
*
MQMD_LENGTH             EQU  *-MQMD
                        ORG  MQMD
MQMD_AREA               DS   CL(MQMD_LENGTH)
```

Deklaration in Visual Basic

```
Type MQMD
  StrucId          As String*4  'Structure identifier'
  Version          As Long      'Structure version number'
  Report           As Long      'Options for report messages'
  MsgType          As Long      'Message type'
  Expiry           As Long      'Message lifetime'
  Feedback         As Long      'Feedback or reason code'
  Encoding         As Long      'Numeric encoding of message data'
  CodedCharSetId   As Long      'Character set identifier of message'
                                'data'
  Format           As String*8  'Format name of message data'
  Priority         As Long      'Message priority'
  Persistence      As Long      'Message persistence'
  MsgId            As MQBYTE24  'Message identifier'
  CorrelId         As MQBYTE24  'Correlation identifier'
  BackoutCount     As Long      'Backout counter'
  ReplyToQ         As String*48 'Name of reply queue'
  ReplyToQMgr      As String*48 'Name of reply queue manager'
  UserIdentifier   As String*12 'User identifier'
  AccountingToken  As MQBYTE32  'Accounting token'
  ApplIdentityData As String*32 'Application data relating to identity'
  PutApplType      As Long      'Type of application that put the'
                                'message'
  PutApplName      As String*28 'Name of application that put the'
                                'message'
  PutDate          As String*8  'Date when message was put'
  PutTime          As String*8  'Time when message was put'
  ApplOriginData   As String*4  'Application data relating to origin'
  GroupId          As MQBYTE24  'Group identifier'
  MsgSeqNumber     As Long      'Sequence number of logical message'
                                'within group'
  Offset           As Long      'Offset of data in physical message'
                                'from start of logical message'
  MsgFlags         As Long      'Message flags'
  OriginalLength   As Long      'Length of original message'
End Type
```

MQMDE – Nachrichtendeskriptorerweiterung

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 516. Felder in der MQMDE

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId

Tabelle 516. Felder in der MQMDE (Forts.)		
Feld	**Beschreibung**	**Thema**
Version	Strukturversionsnummer	Version
StrucLength	Länge der MQMDE-Struktur	StrucLength
Encoding	Numerische Codierung der Daten, die auf MQMDE folgen.	Encoding
CodedCharSetId	Zeichensatzkennung von Daten, die auf MQMDE folgen.	CodedCharSetId
Format	Formatname von Daten, die auf MQMDE folgen	Format
Flags	Allgemeine Flags	Flags
GroupId	Gruppen-ID	GroupId
MsgSeqNumber	Folgenummer einer logischen Nachricht innerhalb einer Gruppe	MsgSeqNumber
Offset	Relative Adresse von Daten in einer physischen Nachricht ab dem Anfang der logischen Nachricht	Offset
MsgFlags	Nachrichtenmarkierungen	MsgFlags
OriginalLength	Länge der ursprünglichen Nachricht	OriginalLength

Überblick für MQMDE

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQMDE-Struktur beschreibt die Daten, die manchmal vor den Anwendungsnachrichtendaten auftreten. Die Struktur enthält diejenigen MQMD-Felder, die in MQMD Version-2, aber nicht in MQMD Version-1 vorliegen.

Formatname: MQFMT_MD_EXTENSION.

Zeichensatz und Codierung: Daten in der MQMDE müssen dem Zeichensatz und der Codierung des lokalen Warteschlangenmanagers entsprechen; diese werden von dem Warteschlangenmanagerattribut *CodedCharSetId* und für die Programmiersprache C von MQENC_NATIVE vorgegeben.

Definieren Sie den Zeichensatz und die Codierung von MQMDE in den Feldern *CodedCharSetId* und *Encoding* von:

- dem MQMD (wenn die MQMDE-Struktur am Anfang der Nachrichtendaten steht), oder
- der Header-Struktur, die der MQMDE-Struktur vorausgeht (alle anderen Fälle).

Wenn Zeichensatz und Codierung der MQMDE nicht dem Zeichensatz und der Codierung des Warteschlangenmanagers entspricht, wird die MQMDE akzeptiert, aber nicht berücksichtigt; die MQMDE wird also wie Nachrichtendaten behandelt.

Anmerkung: Verwenden Sie bei mit Micro Focus COBOL kompilierten Windows-Anwendungen einen Wert von MQENC_NATIVE, der sich von der Codierung des Warteschlangenmanagers unterscheidet. Obwohl numerische Felder in der MQMD-Struktur in den MQPUT-, MQPUT1- und MQGET-Aufrufen die Codierung von Micro Focus COBOL besitzen müssen, müssen numerische Felder in der MQMDE-Struktur die Codierung des Warteschlangenmanagers besitzen. Letzteres wird für die Programmiersprache C von MQENC_NATIVE angegeben und hat den Wert 546.

Verwendung: Bei Anwendungen, die einen MQMD Version-2 verwenden, tritt keine MQMDE-Struktur auf. Bei Fachanwendungen jedoch sowie bei Anwendungen, die weiterhin MQMD Version-1 verwenden,

tritt eine MQMDE eventuell in einigen Situationen auf. Die MQMDE-Struktur kann unter den folgenden Umständen auftreten:

- Bei den MQPUT- und MQPUT1-Aufrufen mit angegeben
- Vom MQGET-Aufruf zurückgegeben
- In Nachrichten innerhalb von Übertragungswarteschlangen

MQMDE angegeben in MQPUT- und MQPUT1-Aufrufen: Wenn die Anwendung in MQPUT- und MQPUT1-Aufrufen einen Version-1 MQMD bereitstellt, besteht für die Anwendung die Möglichkeit, den Nachrichtendaten eine MQMDE voranzustellen, indem das Feld *Format* im MQMD auf MQFMT_MD_EXTENSION gesetzt wird, um anzuzeigen, dass MQMDE vorhanden ist. Wenn die Anwendung keine MQMDE angibt, setzt der Warteschlangenmanager Standardwerte für die Felder in der MQMDE voraus. Die vom Warteschlangenmanager verwendeten Standardwerte entsprechen den Anfangswerten der Struktur; weitere Informationen finden Sie in Tabelle 518 auf Seite 458.

Wenn die Anwendung einen MQMD Version-2 bereitstellt *und* den Anwendungsnachrichtendaten eine MQMDE voranstellt, werden die Strukturen so verarbeitet wie in Tabelle 517 auf Seite 455 angegeben.

Tabelle 517. Aktionen des Warteschlangenmanagers, wenn MQMDE auf MQPUT oder MQPUT1 für MQMDE festgelegt wird

MQMD-Version	Werte von Feldern der Version 2	Werte entsprechender Felder in der MQMDE	Vom Warteschlangenmanager durchgeführte Aktion
1	-	gültig sind	MQMDE wird berücksichtigt
2	Standard	gültig sind	MQMDE wird berücksichtigt
2	Kein Standard	gültig sind	MQMDE wird wie Nachrichtendaten behandelt
1 oder 2	Alle	Ungültig	Aufruf schlägt mit entsprechendem Ursachencode fehl
1 oder 2	Alle	Falscher Zeichensatz oder falsche Codierung der MQMDE oder die MQMDE-Version wird nicht unterstützt	MQMDE wird wie Nachrichtendaten behandelt

Anmerkung: Wenn unter z/OS die Anwendung einen MQMD Version-1 mit einer MQMDE angibt, wertet der Warteschlangenmanager die MQMDE nur dann aus, wenn der *IndexType* der Warteschlange MQIT_GROUP_ID entspricht.

Es gibt einen Sonderfall. Wenn die Anwendung einen MQMD Version-2 verwendet, um eine Nachricht einzureihen, die ein Segment ist (d. h. bei der entweder das Flag MQMF_SEGMENT oder das Flag MQMF_LAST_SEGMENT gesetzt ist), und wenn der Formatname im MQMD MQFMT_DEAD_LETTER_HEADER ist, generiert der Warteschlangenmanager eine MQMDE-Struktur und fügt sie *zwischen* der MQDLH-Struktur und den nachfolgenden Daten ein. In dem MQMD, den der Warteschlangenmanager mit der Nachricht beibehält, werden die Version-2-Felder auf ihre Standardwerte gesetzt.

Einige der Felder, die im MQMD Version-2 vorliegen, aber nicht im MQMD Version-1, sind bei MQPUT und MQPUT1 Ein-/Ausgabefelder. Der Warteschlangenmanager gibt jedoch als Ausgabe der MQPUT- und MQPUT1- Aufrufe *keine* Werte in den entsprechenden Feldern in der MQMDE zurück. Wenn für die Anwendung diese Ausgabewerte erforderlich sind, muss ein MQMD Version-2 vorliegen.

MQMDE zurückgegeben von einem MQGET-Aufruf: Wenn die Anwendung bei einem MQGET-Aufruf einen MQMD Version-1 bereitstellt, stellt der Warteschlangenmanager der Nachricht eine MQMDE voran. Dies geschieht jedoch nur dann, wenn mindestens ein Feld in der MQMDE keinen Standardwert hat. Der Warteschlangenmanager setzt in MQMD das Feld *Format* auf MQFMT_MD_EXTENSION, um damit anzugeben, dass eine MQMDE vorhanden ist.

Wenn die Anwendung am Anfang des Parameters *Buffer* eine MQMDE bereitstellt, wird die MQMDE ignoriert. Bei Rückgabe eines MQGET-Aufrufs wird er von der MQMDE durch die Nachricht (falls erforderlich) ersetzt oder von den Anwendungsnachrichtendaten überschrieben (falls die MQMDE nicht erforderlich ist).

Wenn der MQGET-Aufruf eine MQMDE zurückgibt, entsprechen die Daten in MQMDE normalerweise dem Zeichensatz und der Codierung des Warteschlangenmanagers. Allerdings kann die MQMDE in den nachfolgend aufgeführten Fällen über einen anderen Zeichensatz oder eine andere Codierung verfügen:

- Die MQMDE wurde beim MQPUT- oder beim MQPUT1-Aufruf wie Daten behandelt (Tabelle 517 auf Seite 455 gibt an, unter welchen Umständen es dazu kommen kann).
- Die Nachricht wurde von einem fernen Warteschlangenmanager empfangen, der über eine TCP-Verbindung verbunden ist, und der empfangende Nachrichtenkanalagent (MCA) wurde nicht ordnungsgemäß eingerichtet.

Anmerkung: Verwenden Sie bei mit Micro Focus COBOL kompilierten Windows-Anwendungen einen Wert von MQENC_NATIVE, der sich von der Codierung des Warteschlangenmanagers unterscheidet (siehe oben).

MQMDE bei Nachrichten in Übertragungswarteschlangen: Nachrichten in Übertragungswarteschlangen wird die MQXQH-Struktur vorangestellt, in der ein MQMD Version-1 enthalten ist. Eine MQMDE kann ebenfalls vorhanden und zwischen der MQXQH-Struktur und den Anwendungsnachrichtendaten positioniert sein, aber normalerweise ist sie nur vorhanden, wenn mindestens eines der Felder in der MQMDE keinen Standardwert hat.

Andere MQ-Headerstrukturen können zwischen der MQXQH-Struktur und den Anwendungsnachrichtendaten ebenfalls vorkommen. Zum Beispiel ist, wenn der nicht zustellbare Header MQDLH vorhanden ist und die Nachricht kein Segment darstellt, die Reihenfolge folgendermaßen:

- MQXQH (mit einem MQMD Version-1)
- MQMDE
- MQDLH
- Anwendungsnachrichtendaten

Felder für MQMDE

Die MQMDE-Struktur enthält die folgenden Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)

Dies gibt die Zeichensatzkennung der der MQMDE-Struktur folgenden Daten an; es wird nicht auf Zeichendaten in der MQMDE-Struktur selbst angewendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Der Warteschlangenmanager prüft nicht, ob das Feld gültig ist. Folgende Sonderwerte sind zulässig:

MQCCSI_INHERIT
Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

MQCCSI_INHERIT kann nicht verwendet werden, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

Dieser Wert wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowsund WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

Encoding (MQLONG)

Dies gibt die numerische Codierung der der MQMDE-Struktur folgenden Daten an; es wird nicht auf numerische Daten in der MQMDE-Struktur selbst angewendet.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Der Warteschlangenmanager prüft nicht, ob das Feld gültig ist. Weitere Informationen zu Datencodierungen finden Sie in der Beschreibung des *Encoding*-Felds in „MQMD - Nachrichtendeskriptor" auf Seite 399.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

Flags (MQLONG)

Das folgende Flag kann angegeben werden:

MQMDEF_NONE
Keine Flags.

Der Anfangswert dieses Felds ist MQMDEF_NONE.

Format (MQCHAR8)

In diesem Feld wird der Formatname der Daten angegeben, die der MQMDE-Struktur folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Der Warteschlangenmanager prüft nicht, ob das Feld gültig ist. Weitere Informationen zu Formatnamen finden Sie in der Beschreibung des *Format*-Feldes in „MQMD - Nachrichtendeskriptor" auf Seite 399.

Der Anfangswert dieses Felds ist MQFMT_NONE.

GroupId (MQBYTE24)

Weitere Informationen finden Sie in der Beschreibung des Feldes *GroupId* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399. Der Anfangswert dieses Felds ist MQGI_NONE.

MsgFlags (MQLONG)

Weitere Informationen finden Sie in der Beschreibung des Feldes *MsgFlags* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399. Der Anfangswert dieses Felds ist MQMF_NONE.

MsgSeqNumber (MQLONG)

Weitere Informationen finden Sie in der Beschreibung des Feldes *MsgSeqNumber* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399. Der Anfangswert dieses Felds ist 1.

Offset (MQLONG)

Weitere Informationen finden Sie in der Beschreibung des Feldes *Offset* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399. Der Anfangswert dieses Feldes ist 0.

OriginalLength (MQLONG)

Weitere Informationen finden Sie in der Beschreibung des Feldes *OriginalLength* im Abschnitt „MQMD - Nachrichtendeskriptor" auf Seite 399. Der Anfangswert dieses Felds ist MQOL_UNDEFINED.

StrucId (MQCHAR4)

Folgende Werte sind möglich:

MQMDE_STRUC_ID
ID für die Struktur der Nachrichtendeskriptorerweiterung.

Für die Programmiersprache C ist auch die Konstante MQMDE_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQMDE_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQMDE_STRUC_ID.

StrucLength (MQLONG)

Dies gibt die Länge der MQMDE-Struktur an; es wird der folgende Wert definiert:

MQMDE_LENGTH_2
Länge der Struktur der Nachrichtendeskriptorerweiterung Version-2.

Der Anfangswert dieses Felds ist MQMDE_LENGTH_2.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQMDE_VERSION_2
Version-2 Struktur der Nachrichtendeskriptorerweiterung.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQMDE_CURRENT_VERSION
Aktuelle Version der Struktur Nachrichtendeskriptorerweiterung.

Der Anfangswert dieses Felds ist MQMDE_VERSION_2.

Anfangswerte und Sprachendeklarationen für MQMDE

Tabelle 518. Anfangswerte von Feldern in MQMDE für MQMDE

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQMDE_STRUC_ID	'MDE¬'
Version	MQMDE_VERSION_2	2
StrucLength	MQMDE_LENGTH_2	72
Encoding	MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQMDEF_NONE	0
GroupId	MQGI_NONE	Nullen
MsgSeqNumber	--	1
Offset	--	0
MsgFlags	MQMF_NONE	0
OriginalLength	MQOL_UNDEFINED	-1

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQMDE_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQMDE structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQMDE */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQMDE */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQMDE */
  MQLONG    Flags;           /* General flags */
  MQBYTE24  GroupId;         /* Group identifier */
  MQLONG    MsgSeqNumber;    /* Sequence number of logical message
                                within group */
  MQLONG    Offset;          /* Offset of data in physical message from
                                start of logical message */
  MQLONG    MsgFlags;        /* Message flags */
  MQLONG    OriginalLength;  /* Length of original message */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQMDE based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQMDE structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQMDE */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQMDE */
  3 Format         char(8),       /* Format name of data that follows
                                     MQMDE */
  3 Flags          fixed bin(31), /* General flags */
  3 GroupId        char(24),      /* Group identifier */
  3 MsgSeqNumber   fixed bin(31), /* Sequence number of logical message
                                     within group */
  3 Offset         fixed bin(31), /* Offset of data in physical message
                                     from start of logical message */
  3 MsgFlags       fixed bin(31), /* Message flags */
  3 OriginalLength fixed bin(31); /* Length of original message */
```

Deklaration in High Level Assembler

```
MQMDE                 DSECT
MQMDE_STRUCID         DS   CL4   Structure identifier
MQMDE_VERSION         DS   F     Structure version number
MQMDE_STRUCLENGTH     DS   F     Length of MQMDE structure
MQMDE_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQMDE
MQMDE_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQMDE
MQMDE_FORMAT          DS   CL8   Format name of data that follows MQMDE
MQMDE_FLAGS           DS   F     General flags
MQMDE_GROUPID         DS   XL24  Group identifier
MQMDE_MSGSEQNUMBER    DS   F     Sequence number of logical message
*                                within group
MQMDE_OFFSET          DS   F     Offset of data in physical message from
*                                start of logical message
MQMDE_MSGFLAGS        DS   F     Message flags
MQMDE_ORIGINALLENGTH  DS   F     Length of original message
*
MQMDE_LENGTH          EQU  *-MQMDE
                      ORG  MQMDE
MQMDE_AREA            DS   CL(MQMDE_LENGTH)
```

Deklaration in Visual Basic

```
Type MQMDE
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQMDE structure'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'MQMDE'
  CodedCharSetId As Long     'Character-set identifier of data that'
                             'follows MQMDE'
  Format         As String*8 'Format name of data that follows MQMDE'
  Flags          As Long     'General flags'
  GroupId        As MQBYTE24 'Group identifier'
  MsgSeqNumber   As Long     'Sequence number of logical message within'
                             'group'
  Offset         As Long     'Offset of data in physical message from'
                             'start of logical message'
  MsgFlags       As Long     'Message flags'
  OriginalLength As Long     'Length of original message'
End Type
```

MQMHBO – Nachrichtenhandle-zu-Puffer-Optionen

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur. MQMHBO-Struktur-Nachrichtenkennung zu Pufferoptionen

Tabelle 519. Felder in MQMHBO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQMHBUF	Options

Überblick für MQMHBO

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ MQI-Clients.

Zweck: Mit der MQMHBO-Struktur können Anwendungen Optionen festlegen, die steuern, wie Puffer aus Nachrichtenkennungen erzeugt werden. Bei der Struktur handelt es sich um einen Eingabeparameter im MQMHBUF-Aufruf.

Zeichensatz und Codierung: Die Daten in MQMHBO müssen im Zeichensatz und in der Codierung der Anwendung (MQENC_NATIVE) enthalten sein.

Felder für MQMHBO

Optionsstruktur Nachrichtenkennung für Puffer - Felder

Die MQMHBO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)
Optionsstruktur Nachrichtenkennung für Puffer - Feld Options

Diese Optionen steuern die Aktion von MQMHBUF.

Sie müssen die folgende Option angeben:

MQMHBO_PROPERTIES_IN_MQRFH2
Bei der Umwandlung von Eigenschaften von einer Nachrichtenkennung in einen Puffer wandeln Sie sie in das Format MQRFH2 um.

Optional können Sie auch folgenden Wert angeben. Je nach Bedarf können Werte:

- gemeinsam hinzugefügt werden (dieselbe Konstante nicht mehr als einmal hinzufügen) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

MQMHBO_DELETE_PROPERTIES
Eigenschaften, die zum Puffer hinzugefügt werden, werden aus der Nachrichtenkennung gelöscht. Schlägt der Aufruf fehl, werden keine Eigenschaften gelöscht.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQMHBO_PROPERTIES_IN_MQRFH2.

StrucId (MQCHAR4)
Optionsstruktur der Nachrichtenkennung für Puffer - Feld StrucId

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQMHBO_STRUC_ID
ID der Struktur von Nachrichtenkennung-zu-Puffer-Optionen.

Für die Programmiersprache C wird auch die Konstante MQMHBO_STRUC_ID_ARRAY definiert; diese hat denselben Wert wie MQMHBO_STRUC_ID, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQMHBO_STRUC_ID.

Version (MQLONG)
Optionsstruktur Nachrichtenkennung für Puffer - Feld Version

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQMHBO_VERSION_1
Versionsnummer der Struktur von Nachrichtenkennung-zu-Puffer-Optionen.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQMHBO_CURRENT_VERSION
Aktuelle Version der Struktur von Nachrichtenkennung-zu-Puffer-Optionen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQMHBO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQMHBO

Struktur Nachrichtenkennung an Puffer - Anfangswerte

Tabelle 520. Anfangswerte der Felder in MQMHBO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQMHBO_STRUC_ID	'MHBO'

Tabelle 520. Anfangswerte der Felder in MQMHBO (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
Version	MQMHBO_VERSION_1	1
Options	MQMHBO_PROPERTIES_IN_MQRFH2	

Anmerkungen:

1. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
2. In der Programmiersprache C enthält die MakrovariableMQMHBO_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

Deklaration in Programmiersprache C
Optionsstruktur Nachrichtenkennung für Puffer - Deklaration für Programmiersprache C

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQMHBUF */
};
```

COBOL-Delaration
Optionsstruktur Nachrichtenkennung für Puffer - Deklaration für Programmiersprache COBOL

```
**   MQMHBO structure
  10 MQMHBO.
**    Structure identifier
   15 MQMHBO-STRUCID               PIC X(4).
**    Structure version number
   15 MQMHBO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQMHBUF
   15 MQMHBO-OPTIONS               PIC S9(9) BINARY.
```

Deklaration in PL/I
Optionsstruktur Nachrichtenkennung für Puffer - Deklaration für Programmiersprache PL/I

```
Dcl
 1 MQMHBO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQMHBUF */
```

Deklaration in High Level Assembler
Optionsstruktur Nachrichtenkennung für Puffer - Deklaration für Programmiersprache Assembler

```
MQMHBO                  DSECT
MQMHBO_STRUCID          DS   CL4  Structure identifier
MQMHBO_VERSION          DS   F    Structure version number
MQMHBO_OPTIONS          DS   F    Options that control the
*                                  action of MQMHBUF
MQMHBO_LENGTH           EQU  *-MQMHBO
MQMHBO_AREA             DS   CL(MQMHBO_LENGTH)
```

MQOD - Objektdeskriptor

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
ObjectType	Objekttyp	ObjectType
ObjectName	Objektname	ObjectName
ObjectQMgrName	Name des Objektwarteschlangenmanagers	ObjectQMgrName
DynamicQName	Name der dynamischen Warteschlange	DynamicQName
AlternateUserId	Alternative Benutzer-ID	AlternateUserId
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQOD_VERSION_2 ist.		
RecsPresent	Anzahl der vorhandenen Objektdatensätze	RecsPresent
KnownDestCount	Anzahl der lokalen Warteschlangen, die erfolgreich geöffnet wurden	KnownDestCount
UnknownDestCount	Anzahl der fernen Warteschlangen, die erfolgreich geöffnet wurden	UnknownDestCount
InvalidDestCount	Anzahl der Warteschlangen, die nicht geöffnet werden konnten	InvalidDestCount
ObjectRecOffset	Relative Adresse des ersten Objektdatensatzes ab dem Anfang des MQOD	ObjectRecOffset
ResponseRecOffset	Relative Adresse des ersten Antwortdatensatzes ab dem Anfang des MQOD	ResponseRecOffset
ObjectRecPtr	Adresse des ersten Objektdatensatzes	ObjectRecPtr
ResponseRecPtr	Adresse des ersten Antwortdatensatzes	ResponseRecPtr
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQOD_VERSION_3 ist.		
AlternateSecurityId	Alternative Sicherheits-ID	AlternateSecurityId
ResolvedQName	Aufgelöster Warteschlangenname	ResolvedQName
ResolvedQMgrName	Aufgelöster Name des Warteschlangenmanagers	ResolvedQMgrName
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQOD_VERSION_4 ist.		
ObjectString	Langer Objektname	ObjectString
SelectionString	Auswahlzeichenfolge	SelectionString
ResObjectString	Aufgelöster langer Objektname	ResObjectString
ResolvedType	Aufgelöster Objekttyp	ResolvedType

Überblick für MQOD

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ-MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQOD-Struktur wird verwendet, um ein Objekt namentlich anzugeben. Folgende Objekttypen sind gültig:

- Warteschlange oder Verteilerliste
- Namensliste
- Prozessdefinition
- Warteschlangenmanager
- Thema

Die Struktur ist ein Ein-/Ausgabeparameter in den Aufrufen MQOPEN und MQPUT1.

Version: Die aktuelle MQOD-Version ist MQOD_VERSION_4. Anwendungen, die auf mehrere Umgebungen portierbar sein sollen, müssen sicherstellen, dass die erforderliche Version von MQOD von allen betroffenen Umgebungen unterstützt wird. Felder, die nur in den neueren Versionen der Struktur vorhanden sind, werden in den nachfolgenden Beschreibungen entsprechend gekennzeichnet.

Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützten Programmiersprachen bereitgestellt werden, enthalten die aktuellste Version von MQOD, die von der Umgebung unterstützt wird, wobei der Anfangswert des Feldes *Version* auf MQOD_VERSION_1 gesetzt ist. Um Felder zu verwenden, die in der Struktur der Version 1 nicht verfügbar sind, muss die Anwendung das Feld *Version* auf die Versionsnummer der erforderlichen Version setzen.

Um eine Verteilerliste zu öffnen, muss die *Version* MQOD_VERSION_2 oder höher sein.

Zeichensatz und Codierung: Die Daten in der MQOD müssen dem Zeichensatz entsprechend, der dem *CodedCharSetId*-Attribut des Warteschlangenmanagers entspricht, und der Codierung des lokalen Warteschlangenmanagers, die von MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQOD

Die MQOD-Struktur enthält die nachfolgend aufgeführten Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

AlternateSecurityId (MQBYTE40)

Dies ist eine Sicherheits-ID, die zusammen mit der *AlternateUserId* an den Berechtigungsservice übermittelt wird, damit geeignete Berechtigungsprüfungen ausgeführt werden können. *AlternateSecurityId* wird nur in den folgenden Fällen verwendet:

- MQOO_ALTERNATE_USER_AUTHORITY wird bei dem MQOPEN-Aufruf angegeben oder
- MQPMO_ALTERNATE_USER_AUTHORITY wird bei dem MQPUT1-Aufruf angegeben

und das *AlternateUserId*-Feld ist bis zum ersten Nullzeichen oder bis zum Ende des Felds nicht vollständig leer.

Bei Windows kann *AlternateSecurityId* verwendet werden, um die Sicherheits-ID (SID) von Windows bereitzustellen, die die *AlternateUserId* eindeutig definiert. Die SID für einen Benutzer kann über den API-Aufruf `LookupAccountName()` Windows vom Windows -System abgerufen werden.

Unter z/OS wird dieses Feld ignoriert.

Das *AlternateSecurityId*-Feld hat die folgende Struktur:

- Das erste Byte ist eine binäre Ganzzahl zur Angabe der Länge der nachfolgenden signifikanten Daten. Das Längenbyte selbst wird bei diesem Wert nicht mit berücksichtigt. Wenn keine Sicherheits-ID vorhanden ist, beträgt die Länge null.
- Das zweite Byte gibt die Art der vorhandenen Sicherheits-ID an. Die folgenden Werte sind möglich:

  MQSIDT_NT_SECURITY_ID
  Windows-Sicherheits-ID.

  MQSIDT_NONE
  Keine Sicherheits-ID.

- Das dritte Byte und die darauf folgenden Bytes bis zur der vom ersten Byte definierten Länge enthalten die Sicherheits-ID selbst.
- Die weiteren Byte im Feld werden auf binär null festgelegt.

Sie können den folgenden Spezialwert verwenden:

MQSID_NONE
Es ist keine Sicherheits-ID angegeben.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQSID_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQSID_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Eingabefeld. Die Länge dieses Felds ist durch MQ_SECURITY_ID_LENGTH vorgegeben. Der Anfangswert dieses Felds ist MQSID_NONE. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_3.

AlternateUserId (MQCHAR12)

Wenn Sie für den MQOPEN-Aufruf MQOO_ALTERNATE_USER_AUTHORITY oder MQPMO_ALTERNATE_USER_AUTHORITY für den MQPUT1-Aufruf angeben, enthält dieses Feld eine alternative Benutzer-ID, die statt der Benutzer-ID verwendet wird, unter der die Anwendung aktuell ausgeführt wird, um die Berechtigung für das Öffnen zu überprüfen. Einige Prüfungen, beispielsweise Kontextprüfungen, werden jedoch nach wie vor mit der aktuellen Benutzer-ID ausgeführt.

Wenn MQOO_ALTERNATE_USER_AUTHORITY oder MQPMO_ALTERNATE_USER_AUTHORITY angegeben werden und dieses Feld bis zum ersten Nullzeichen oder bis zum Ende des Felds vollständig leer ist, kann das Öffnen nur erfolgreich ausgeführt werden, wenn keine Benutzerberechtigung benötigt wird, um das Objekt mit den angegebenen Optionen zu öffnen.

Wenn weder MQOO_ALTERNATE_USER_AUTHORITY noch MQPMO_ALTERNATE_USER_AUTHORITY angegeben ist, wird dieses Feld ignoriert.

Für die angegebenen Umgebungen gelten die folgenden Unterschiede:

- Bei z/OS werden nur die ersten 8 Zeichen von *AlternateUserId* verwendet, um die Berechtigung für das Öffnen zu überprüfen. Die aktuelle Benutzer-ID muss jedoch zur Angabe dieser bestimmten alternativen Benutzer-ID berechtigt sein. Alle 12 Zeichen der alternativen Benutzer-ID werden für diese Prüfung verwendet. Die Benutzer-ID darf nur vom externen Sicherheitsmanager erlaubte Zeichen enthalten.

  Wenn *AlternateUserId* für eine Warteschlange angegeben wird, kann der Wert anschließend vom Warteschlangenmanager beim Einreihen von Nachrichten verwendet werden. Wenn die beim MQPUT- oder MQPUT1-Aufruf angegebenen MQPMO_*_CONTEXT-Optionen den Warteschlangenmanager dazu veranlassen, die Identitätskontextinformationen zu generieren, platziert der Warteschlangenmanager anstelle der aktuellen Benutzer-ID die *AlternateUserId* im *UserIdentifier*-Feld des MQMD der Nachricht.
- In anderen Umgebungen wird *AlternateUserId* nur für Zugriffssteuerungsprüfungen des Objekts verwendet, das geöffnet wird. Wenn es sich bei dem Objekt um eine Warteschlange handelt, wirkt sich die *AlternateUserId* nicht auf den Inhalt des *UserIdentifier*-Felds in den MQMD der Nachrichten aus, die mit dieser Warteschlangenkennung gesendet werden.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 12 leere Zeichen in anderen Programmiersprachen.

DynamicQName (MQCHAR48)

Dies ist der Name einer dynamischen Warteschlange, die von dem MQOPEN-Aufruf erstellt werden soll. Dies ist nur dann relevant, wenn *ObjectName* den Namen der Modellwarteschlange angibt; in allen anderen Fällen wird *DynamicQName* ignoriert.

Die Charaktere, die für den Namen gültig sind, entsprechen denen von *ObjectName*, nur ist ein Stern ebenfalls gültig. Ein Name, der einem Leerzeichen entspricht, bzw. ein Name, in dem vor dem ersten Nullzeichen nur Leerzeichen auftreten, ist nicht gültig, wenn *ObjectName* der Name einer Modellwarteschlange ist.

Falls es sich bei dem letzten Zeichen des Namens, das kein Leerzeichen ist, um einen Stern handelt (*), ersetzt der Warteschlangenmanager den Stern mit einer Zeichenfolge, die sicherstellt, dass der für die Warteschlange generierte Name auf dem lokalen Warteschlangenmanager eindeutig ist. Damit hierfür eine ausreichende Anzahl an erlaubten Zeichen zur Verfügung steht, ist der Stern nur an den Positionen 1 bis 33 gültig. Auf den Stern dürfen nur Leerzeichen oder Nullzeichen folgen.

Der Stern darf das erste Zeichen der Zeichenfolge sein. In diesem Fall besteht der Name ausschließlich aus den von dem Warteschlangenmanager erzeugten Zeichen.

Verwenden Sie bei z/OS keinen Namen, dessen erstes Zeichen ein Stern ist, da keine Sicherheitsprüfung einer Warteschlange durchgeführt werden kann, deren vollständiger Name automatisch erzeugt wurde.

Dies ist ein Eingabefeld. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Felds wird von der Umgebung bestimmt:

- Unter z/OSlautet der Wert `'CSQ.*'`.
- Auf anderen Plattformen lautet der Wert `'AMQ.*'`.

Der Wert ist in der Programmiersprache C eine auf null endende Zeichenfolge und in anderen Programmiersprachen eine mit Leerzeichen aufgefüllte Zeichenfolge.

InvalidDestCount (MQLONG)

Dies ist die Anzahl der Warteschlangen, die nicht erfolgreich geöffnet werden konnten. Wenn es vorhanden ist, wird dieses Feld auch gesetzt, wenn eine einzelne Warteschlange geöffnet wird, die nicht zu einer Verteilerliste gehört.

Anmerkung: Wenn es vorhanden ist, wird dieses Feld *nur* gesetzt, wenn der *CompCode*-Parameter beim MQOPEN- oder beim MQPUT1-Aufruf MQCC_OK oder MQCC_WARNING ist. Es wird *nicht* gesetzt, wenn der *CompCode*-Parameter MQCC_FAILED ist.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQOD_VERSION_1ist.

KnownDestCount (MQLONG)

Dies ist die Anzahl der als lokale Warteschlangen aufgelösten und erfolgreich geöffneten Warteschlangen in der Verteilerliste. Nicht in dieser Anzahl enthalten sind Warteschlangen, die als ferne Warteschlangen aufgelöst sind (auch wenn zur Abspeicherung der Nachricht zunächst eine lokale Übertragungswarteschlange verwendet wird). Wenn es vorhanden ist, wird dieses Feld auch gesetzt, wenn eine einzelne Warteschlange geöffnet wird, die nicht zu einer Verteilerliste gehört.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQOD_VERSION_1ist.

ObjectName (MQCHAR48)

Dies ist der lokale Name des Objekts, wie er auf dem Warteschlangenmanager durch *ObjectQMgrName* definiert wird. Der Name kann folgende Zeichen enthalten:

- Großbuchstaben (A bis Z)
- Kleinbuchstaben des Alphabets(a bis z)
- Numerische Ziffern (0 bis 9)
- Punkt (.), Schrägstrich (/), Unterstrich (_), Prozent (%)

Der Name darf keine führenden oder eingebetteten Leerzeichen enthalten, wohl aber abschließende Leerzeichen. Verwenden Sie ein Nullzeichen zur Kennzeichnung des Endes signifikanter Daten im Namen; die Null und alle ihr folgenden Zeichen werden wie Leerzeichen behandelt. Die folgenden Beschränkungen gelten für die angegebenen Umgebungen:

- Bei Systemen, die EBCDIC Katakana verwenden, können keine Kleinbuchstaben verwendet werden.
- Unter z/OS:
  - Vermeiden Sie Namen, die mit einem Unterstrich beginnen oder enden; sie können von den Betriebs- und Steuerkonsolen nicht verarbeitet werden.
  - Das Prozentzeichen hat eine spezielle Bedeutung für RACF. Wenn RACF als externer Sicherheitsmanager verwendet wird, dürfen Namen kein Prozentzeichen enthalten. Andernfalls werden diese Namen nicht in Sicherheitsprüfungen einbezogen, wenn generische RACF-Profile verwendet werden.
- In IBM i müssen innerhalb von Befehlen vorkommende Namen, die Kleinbuchstaben, Schrägstriche oder Prozentzeichen enthalten, zwischen An- und Abführungszeichen stehen. Diese Anführungszeichen dürfen nicht bei Namen angegeben werden, die Felder in Strukturen oder Parameter bei Aufrufen sind.

Der vollständige Abschnittsname kann aus zwei verschiedenen Feldern erstellt werden: *ObjectName* und *ObjectString*. Ausführliche Informationen über die Verwendung dieser beiden Felder finden Sie im Abschnitt „Verwenden von Themenzeichenfolgen” auf Seite 569.

Die folgenden Punkte finden auf die angegebenen Objekttypen Anwendung:

- Falls *ObjectName* der Name einer Modellwarteschlange ist, erstellt der Warteschlangenmanager eine dynamische Warteschlange mit den Attributen der Modellwarteschlange und gibt im *ObjectName*-Feld den Namen der Warteschlange zurück, die erstellt wurde. Eine Modellwarteschlange kann nur für den MQOPEN-Aufruf angegeben werden- Für den MQPUT1-Aufruf ist sie nicht gültig.
- Wenn *ObjectName* der Name einer Aliaswarteschlange mit TARGTYPE(TOPIC) ist, wird erst eine Sicherheitsprüfung auf der namentlich genannten Aliaswarteschlange ausgeführt; werden Aliaswarteschlangen verwendet, ist dies eine Standardmaßnahme. Wenn die Sicherheitsprüfung erfolgreich abgeschlossen wurde, wird der MQOPEN-Aufruf fortgesetzt und sich wie ein MQOPEN-Aufruf auf einem MQOT_TOPIC verhalten; dies schließt eine Sicherheitsprüfung des Topic-Verwaltungsobjekts mit ein.
- Wenn *ObjectName* und *ObjectQMgrName* eine gemeinsam genutzte Warteschlange ermitteln, die der Gruppe mit gemeinsamer Warteschlange zugehörig ist, zu der der lokale Warteschlangenmanager gehört, darf eine Warteschlangendefinition nicht mit demselben Namen auf dem lokalen Warteschlangenmanager vorhanden sein. Wenn eine solche Definition vorliegt (eine lokale Warteschlange, eine Aliaswarteschlange, eine ferne Warteschlange bzw. eine Modellwarteschlange), schlägt der Aufruf mit dem Ursachencode MQRC_OBJECT_NOT_UNIQUE fehl.
- Wenn es sich bei dem Objekt, das geöffnet wird, um eine Verteilerliste handelt, das heißt, wenn *RecsPresent* größer ist als null, muss *ObjectName* eine Nullzeichenfolge oder leer sein. Wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_OBJECT_NAME_ERROR fehl.
- Wenn *ObjectType* MQOT_Q_MGR ist, gelten besondere Regeln: In diesem Fall muss der Name bis zum ersten Nullzeichen oder dem Ende des Felds vollständig leer sein.

Dies ist ein Ein-/Ausgabefeld für den MQOPEN-Aufruf, wenn *ObjectName* der Name einer Modellwarteschlange ist; in allen anderen Fällen ist es ein Eingabefeld. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ObjectQMgrName (MQCHAR48)

Gibt den Namen des Warteschlangenmanagers an, auf dem das Objekt *ObjectName* definiert ist. Die Zeichen, die für den Namen gültig sind, entsprechen denen für *ObjectName* (siehe Abschnitt „ObjectName (MQCHAR48)” auf Seite 466). Ein Name, der bis zum ersten Nullzeichen oder dem Ende des Felds leer ist, gibt den Warteschlangenmanager an, mit dem die Anwendung verbunden ist, also den lokalen Warteschlangenmanager.

Die folgenden Punkte finden auf die angegebenen Objekttypen Anwendung:

- Wenn *ObjectType* MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS oder MQOT_Q_MGR ist, muss *ObjectQMgrName* der Name des lokalen Warteschlangenmanagers oder leer sein.
- Falls *ObjectName* der Name einer Modellwarteschlange ist, erstellt der Warteschlangenmanager eine dynamische Warteschlange mit den Attributen der Modellwarteschlange und gibt im *ObjectQMgrName*-Feld den Namen des Warteschlangenmanagers zurück, auf dem die Warteschlange erstellt wurde;

dabei handelt es sich um den Namen des lokalen Warteschlangenmanagers. Eine Modellwarteschlange kann nur für den MQOPEN-Aufruf angegeben werden- Für den MQPUT1-Aufruf ist sie nicht gültig.

- Falls *ObjectName* der Name einer Clusterwarteschlange ist und *ObjectQMgrName* leer ist, wird der Bestimmungsort der mit der vom MQOPEN-Aufruf zurückgegebenen Warteschlangenkennung gesendeten Nachrichten vom Warteschlangenmanager (oder vom Exit für Clusterauslastung, falls einer installiert ist) wie folgt ausgewählt:
  - Wenn MQOO_BIND_ON_OPEN angegeben ist, wählt der Warteschlangenmanager während der Verarbeitung des MQOPEN-Aufrufs eine bestimmte Instanz der Clusterwarteschlange aus und sämtliche Nachrichten, die mit dieser Warteschlangenkennung eingereiht werden, werden an diese Instanz gesendet.
  - Wenn MQOO_BIND_NOT_FIXED angegeben ist, kann der Warteschlangenmanager für jeden aufeinanderfolgenden MQPUT-Aufruf, der diese Warteschlangenkennung verwendet, eine andere Instanz der Zielwarteschlange, die sich auf einem anderen Warteschlangenmanager im Cluster befindet, verwenden.

  Wenn die Anwendung eine Nachricht an eine *bestimmte* Instanz einer Clusterwarteschlange, also einer Warteschlangeninstanz, die sich auf einem bestimmten Warteschlangenmanager im Cluster befindet, senden muss, muss die Anwendung den Namen dieses Warteschlangenmanagers im *ObjectQMgrName*-Feld angeben. Dies zwingt den lokalen Warteschlangenmanager dazu, die Nachricht an den angegebenen Ziel-Warteschlangenmanager zu senden.
- Wenn *ObjectName* der Name einer gemeinsam genutzten Warteschlange ist, dieder Gruppe mit gemeinsamer Warteschlange zugehörig ist, zu der der lokale Warteschlangenmanager gehört, kann *ObjectQMgrName* der Name der Gruppe mit gemeinsamer Warteschlange, der Name des lokalen Warteschlangenmanagers oder leer sein; unabhängig davon, welcher dieser Werte angegeben wird, wird die Nachricht in dieselbe Warteschlange eingereiht.

  Gruppen mit gemeinsamer Warteschlange werden nur bei z/OS unterstützt.
- Wenn *ObjectName* der Name einer gemeinsam genutzten Warteschlange ist, die einer fernen Gruppe mit gemeinsamer Warteschlange, also einer Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager *nicht* gehört, zugehörig ist, muss *ObjectQMgrName* der Name der Gruppe mit gemeinsamer Warteschlange sein. Sie können den Namen eines Warteschlangenmanagers benutzen, der zu dieser Gruppe gehört, allerdings kann das die Nachricht verzögern, falls dieser spezielle Warteschlangenmanager nicht verfügbar ist, wenn die Nachricht die Gruppe mit gemeinsamer Warteschlange erreicht.
- Wenn es sich bei dem Objekt, das geöffnet wird, um eine Verteilerliste handelt, das heißt, wenn *RecsPresent* größer ist als null, muss *ObjectQMgrName* eine Nullzeichenfolge oder leer sein. Wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_OBJECT_Q_MGR_NAME_ERROR fehl.

Dies ist ein Ein-/Ausgabefeld für den MQOPEN-Aufruf, wenn *ObjectName* der Name einer Modellwarteschlange ist; in allen anderen Fällen ist es ein Eingabefeld. Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ObjectRecOffset (MQLONG)

Dies ist die in Byte angegebene relative Adresse des ersten MQOR-Objektdatensatzes ab dem Anfang der MQOD-Struktur. Der Offset kann positiv oder negativ sein. *ObjectRecOffset* wird nur verwendet, wenn eine Verteilerliste geöffnet wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Beim Öffnen einer Verteilerliste muss ein Array aus einem oder mehreren MQOR-Objektdatensätzen bereitgestellt werden, um die Namen der Zielwarteschlangen in der Verteilerliste anzugeben. Dies kann auf zwei Arten erfolgen:

- Verwenden Sie das Offset-Feld *ObjectRecOffset*.

  In diesem Fall muss die Anwendung ihre eigene Struktur mit einem MQOD gefolgt von der Feldgruppe der MQOR-Datensätze, die so viele Feldgruppenelemente wie nötig enthalten, deklarieren und *ObjectRecOffset* auf den Offset des ersten Elements in der Feldgruppe vom Anfang des MQOD aus gesehen

setzen. Stellen Sie sicher, dass dieser Offset korrekt ist und dass sein Wert dem Datentyp MQLONG entspricht. Die restriktivste Programmiersprache in diesem Zusammenhang ist COBOL; hier liegt der gültige Bereich zwischen -999.999.999 und +999.999.999.

Verwenden Sie *ObjectRecOffset* für Programmiersprachen, die den Zeigerdatentyp nicht unterstützen oder deren Art der Implementierung des Zeigerdatentyps nicht auf andere Umgebungen übertragbar ist (dies betrifft beispielsweise die Programmiersprache COBOL).

- Verwenden Sie das Zeigerfeld *ObjectRecPtr*.

  In diesem Fall kann die Anwendung die Feldgruppe der MQOR-Strukturen separat von der MQOD-Struktur deklarieren und *ObjectRecPtr* auf die Adresse der Feldgruppe setzen.

  Verwenden Sie *ObjectRecPtr* bei Programmiersprachen, die den Zeigerdatentyp so unterstützen, dass er in andere Umgebungen (beispielsweise in die Programmiersprache C) portierbar ist.

Verwenden Sie unabhängig vom gewählten Verfahren entweder *ObjectRecOffset* oder *ObjectRecPtr* ; der Aufruf schlägt mit dem Ursachencode MQRC_OBJECT_RECORDS_ERROR fehl, wenn beide null oder beide ungleich null sind.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_2.

ObjectRecPtr (MQPTR)

Dies ist die Adresse des ersten MQRR-Objektdatensatzes. *ObjectRecPtr* wird nur verwendet, wenn eine Verteilerliste geöffnet wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Verwenden Sie entweder *ObjectRecPtr* oder *ObjectRecOffset*, um die Objektdatensätze anzugeben, aber verwenden Sie nicht beide; weitere Einzelheiten finden Sie in der Beschreibung des *ObjectRecOffset*-Feldes oben. Wenn Sie *ObjectRecPtr* nicht verwenden, setzen Sie das Feld auf den Nullzeiger oder auf null Bytes.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_2.

Anmerkung: Bei Plattformen, bei denen die Programmiersprache den Datentyp Zeiger nicht unterstützt, wird das Feld als Bytefolge der entsprechenden Länge deklariert, wobei der Anfangswert die vollständig auf null gesetzte Bytefolge ist.

ObjectString (MQCHARV)
Das ObjectString-Feld gibt den langen Objektnamen an.

Dies gibt den zu verwendenden langen Namen an. Dieses Feld wird nur für bestimmte Werte von *ObjectType* referenziert und für alle anderen Werte ignoriert. Weitere Einzelheiten dazu, welche Werte angeben, dass dieses Feld verwendet wird, finden Sie in der Beschreibung von *ObjectType*.

Wenn *ObjectString* nicht ordnungsgemäß und nicht entsprechend der Beschreibung zur Verwendung der MQCHARV-Struktur angegeben wird oder wenn die Zeichenfolge die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_OBJECT_STRING_ERROR fehl.

Dies ist ein Eingabefeld. Die Anfangswerte der Felder in dieser Struktur entsprechen denen der MQCHARV-Struktur.

Der vollständige Abschnittsname kann aus zwei verschiedenen Feldern erstellt werden: *ObjectName* und *ObjectString*. Ausführliche Informationen über die Verwendung dieser beiden Felder finden Sie im Abschnitt „Verwenden von Themenzeichenfolgen” auf Seite 569.

ObjectType (MQLONG)

Der Objekttyp, der im Objektdeskriptor benannt wird. Mögliche Werte:

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal. Der Name des Objekts findet sich im *ObjectName*-Feld.

MQOT_Q
Queue. Der Name des Objekts findet sich im *ObjectName*-Feld.

MQOT_NAMELIST
Namensliste. Der Name des Objekts findet sich im *ObjectName*-Feld.

MQOT_PROCESS
Prozessdefinition. Der Name des Objekts findet sich im *ObjectName*-Feld.

MQOT_Q_MGR
Warteschlangenmanager Der Name des Objekts findet sich im *ObjectName*-Feld.

MQOT_TOPIC
Thema. Der vollständige Abschnittsname kann aus zwei verschiedenen Feldern erstellt werden: *ObjectName* und *ObjectString*.

Einzelheiten zur Verwendung dieser beiden Felder finden Sie in „Verwenden von Themenzeichenfolgen” auf Seite 569.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQOT_Q.

RecsPresent (MQLONG)

Dies ist die Anzahl der durch die Anwendung bereitgestellten MQOR-Objektdatensätze. Wenn diese Zahl größer als null ist, zeigt dies an, dass eine Verteilerliste geöffnet wird, wobei *RecsPresent* der Anzahl der Zielwarteschlangen in der Liste entspricht. Eine Verteilerliste kann auch nur ein einziges Ziel enthalten.

Der Wert von *RecsPresent* darf nicht kleiner als null sein und wenn er größer als null ist, dann muss *ObjectType* MQOT_Q sein; der Aufruf schlägt mit dem Ursachencode MQRC_RECS_PRESENT_ERROR fehl, falls diesen Bedingungen nicht entsprochen wird.

Bei z/OS muss dieses Feld null sein.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_2.

ResObjectString (MQCHARV)
Das ResObjectString-Feld enthält den langen Objektnamen, nachdem der Warteschlangenmanager den Namen, der im *ObjectName*-Feld bereitgestellt wird, aufgelöst hat.

Dieses Feld wird nur für Themen und Aliasnamen von Warteschlangen zurückgegeben, die auf ein Themenobjekt verweisen.

Wenn der ausgeschriebene Objektname in *ObjectString* bereitgestellt wird und im Feld *ObjectName* keine Angaben gemacht werden, ist der in diesem Feld zurückgegebene Wert mit dem in *ObjectString* bereitgestellten Wert identisch.

Wenn dieses Feld ausgeschlossen wird, das heißt, wenn ResObjectString.VSBufSize gleich null ist, wird *ResObjectString* nicht zurückgegeben, sondern in ResObjectString.VSLength wird die Länge zurückgegeben.

Wenn die Puffergröße, die in ResObjectStrng.VSBufSize bereitgestellt wird, kürzer ist als die gesamte *ResObjectString*, wird die Zeichenfolge abgeschnitten und gibt so viele der Zeichen ganz rechts zurück, wie in den bereitgestellten Puffer passen.

Wenn *ResObjectString* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der MQCHARV-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_RES_OBJECT_STRING_ERROR fehl.

ResolvedQMgrName (MQCHAR48)

Dies ist der Name des Zielwarteschlangenmanagers nach der Namensauflösung durch den lokalen Warteschlangenmanager. Der zurückgegebene Name ist der Name des Warteschlangenmanagers, der der Eigner der Warteschlange ist, die von *ResolvedQName* ermittelt wurde. *ResolvedQMgrName* kann der Name des lokalen Warteschlangenmanagers sein.

Wenn *ResolvedQName* eine gemeinsam genutzte Warteschlange ist, deren Eigner die Gruppe mit gemeinsamer Warteschlange ist, zu der der lokale Warteschlangenmanager gehört, dann ist *ResolvedQMgrName* der Name der Gruppe mit gemeinsamer Warteschlange. Wenn die Warteschlange einer anderen Gruppe mit gemeinsamer Warteschlange zugeordnet ist, kann *ResolvedQName* der Name der Gruppe mit gemeinsamer Warteschlange oder der Name eines Warteschlangenmanagers sein, der ein Mitglied der Gruppe mit gemeinsamer Warteschlange ist (die Art des zurückgegebenen Werts ist von den Warteschlangendefinitionen des lokalen Warteschlangenmanagers abhängig).

Ein belegter Wert wird nur dann zurückgegeben, wenn es sich bei dem Objekt um eine einzelne Warteschlange handelt, die zum Durchsuchen, zur Eingabe oder zur Ausgabe oder für eine beliebige Kombination der drei Möglichkeiten geöffnet ist. Wenn auf das geöffnete Objekt eine der folgenden Bedingungen zutrifft, wird *ResolvedQMgrName* auf nicht belegt gesetzt:

- Es ist keine Warteschlange.
- Es ist eine Warteschlange, aber nicht zum Durchsuchen, zur Eingabe oder zur Ausgabe geöffnet.
- Es ist eine Clusterwarteschlange, für die MQOO_BIND_NOT_FIXED angegeben ist (oder MQOO_BIND_AS_Q_DEF ist gültig, wenn das *DefBind*-Warteschlangenattribut den Wert MQBND_BIND_NOT_FIXED hat).
- Es ist eine Verteilerliste.

Dies ist ein Ausgabefeld. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_3.

ResolvedQName (MQCHAR48)

Dies ist der Name der Zielwarteschlange, nachdem der lokale Warteschlangenmanager den Namen aufgelöst hat. Der zurückgegebene Name gibt den Namen einer Warteschlange an, so wie sie auf dem mit *ResolvedQMgrName* angegebenen Warteschlangenmanager definiert ist.

Ein belegter Wert wird nur dann zurückgegeben, wenn es sich bei dem Objekt um eine einzelne Warteschlange handelt, die zum Durchsuchen, zur Eingabe oder zur Ausgabe oder für eine beliebige Kombination der drei Möglichkeiten geöffnet ist. Wenn auf das geöffnete Objekt eine der folgenden Bedingungen zutrifft, wird *ResolvedQName* auf nicht belegt gesetzt:

- Es ist keine Warteschlange.
- Es ist eine Warteschlange, aber nicht zum Durchsuchen, zur Eingabe oder zur Ausgabe geöffnet.
- Es ist eine Verteilerliste.
- Es ist eine Aliaswarteschlange, die ein Themenobjekt referenziert (verweisen Sie stattdessen auf ResObjectString).
- Es ist eine Aliaswarteschlange, die in ein Themenobjekt aufgelöst wird.

Dies ist ein Ausgabefeld. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_3.

ResolvedType (MQLONG)
Der Typ des aufgelösten Objekts (Basisobjekts), das geöffnet wird.

Folgende Werte sind möglich:

MQOT_Q
Das aufgelöste Objekt ist eine Warteschlange. Dieser Wert gilt, wenn eine Warteschlange direkt geöffnet wird oder wenn eine Aliaswarteschlange, die auf eine Warteschlange verweist, geöffnet wird.

MQOT_TOPIC
Das aufgelöste Objekt ist ein Thema. Dieser Wert gilt, wenn ein Thema direkt geöffnet wird oder wenn eine Aliaswarteschlange, die auf ein Themenobjekt verweist, geöffnet wird.

MQOT_NONE
Der aufgelöste Typ ist weder eine Warteschlange noch ein Thema.

ResponseRecOffset (MQLONG)

Dies ist der Offset in Byte des ersten MQRR-Antwortdatensatzes vom Anfang der MQOD-Struktur. Der Offset kann positiv oder negativ sein. *ResponseRecOffset* wird nur verwendet, wenn eine Verteilerliste geöffnet wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Wenn eine Verteilerliste geöffnet wird, können Sie eine Feldgruppe mit mindestens einem MQRR-Antwortdatensatz bereitstellen, um zum einen die Warteschlangen zu ermitteln, bei denen das Öffnen fehlgeschlagen ist (*CompCode*-Feld in MQRR) und zum anderen den Grund für das Fehlschlagen (*Reason*-Feld in MQRR). Die Daten werden in dem Array aus Antwortdatensätzen in der Reihenfolge zurückgegeben, in der die Warteschlangennamen in dem Array aus Objektdatensätzen stehen. Der Warteschlangenmanager setzt nur dann die Antwortdatensätze, wenn das Ergebnis des Aufrufs heterogen ist, d. h. wenn einige Warteschlangen erfolgreich geöffnet wurden, während das Öffnen anderer fehlschlug, bzw. wenn alle Öffnungsversuche fehlschlugen, allerdings aus unterschiedlichen Gründen; der Ursachencode MQRC_MULTIPLE_REASONS des Aufrufs weist darauf hin. Wenn derselbe Ursachencode für alle Warteschlangen zutrifft, wird diese Ursache im *Reason*-Parameter des MQOPEN- oder des MQPUT1-Aufrufs wiedergegeben und die Antwortdatensätze werden nicht gesetzt. Antwortdatensätze sind optional, aber wenn sie zur Verfügung gestellt werden, muss es *RecsPresent* für sie geben.

Die Antwortdatensätze können auf dieselbe Art bereitgestellt werden wie die Objektdatensätze: entweder, indem in *ResponseRecOffset* ein Offset oder indem in *ResponseRecPtr* eine Adresse angegeben wird; weitere Informationen zur Vorgehensweise finden Sie oben in der Beschreibung von *ObjectRecOffset*. Setzen Sie jedoch nur eines der Felder, *ResponseRecOffset* oder *ResponseRecPtr*, auf ungleich null; der Aufruf schlägt mit dem Ursachencode MQRC_RESPONSE_RECORDS_ERROR fehl, wenn beide ungleich null sind.

Beim MQPUT1-Aufruf werden diese Antwortdatensätze dazu verwendet, sowohl Informationen über Fehler zurückzugeben, die auftreten, wenn die Nachricht an Warteschlangen in der Verteilerliste gesendet wird, als auch über Fehler, die beim Öffnen der Warteschlangen auftreten. Der Beendigungs- und der Ursachencode der Put-Operation für eine Warteschlange ersetzen die der Operation zum Öffnen dieser Warteschlange nur dann, wenn der Beendigungscode der letzteren MQCC_OK oder MQCC_WARNING war.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_2.

ResponseRecPtr (MQPTR)

Dies ist die Adresse des ersten MQRR-Antwortdatensatzes. *ResponseRecPtr* wird nur verwendet, wenn eine Verteilerliste geöffnet wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Verwenden Sie entweder *ResponseRecPtr* oder *ResponseRecOffset*, um die Antwortdatensätze anzugeben, aber verwenden Sie nicht beide; weitere Einzelheiten finden Sie in der Beschreibung des *ResponseRecOffset*-Feldes oben. Wenn Sie *ResponseRecPtr* nicht verwenden, setzen Sie das Feld auf den Nullzeiger oder auf null Bytes.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQOD_VERSION_2.

Anmerkung: Bei Plattformen, bei denen die Programmiersprache den Datentyp Zeiger nicht unterstützt, wird das Feld als Bytefolge der entsprechenden Länge deklariert, wobei der Anfangswert die vollständig auf null gesetzte Bytefolge ist.

SelectionString (MQCHARV)
Dies ist die Zeichenfolge, die zum Bereitstellen der Auswahlbedingungen verwendet wird, die beim Abrufen von Nachrichten aus einer Warteschlange verwendet wird.

SelectionString darf in folgenden Fällen nicht bereitgestellt werden:

- Wenn *ObjectType* nicht MQOT_Q ist
- Wenn die Warteschlange, die geöffnet wird, nicht mit einer der MQOO_BROWSE- oder MQOO_INPUT_*-Optionen geöffnet wird

Wenn *SelectionString* in diesen Fällen bereitgestellt wird, schlägt der Aufruf mit dem Ursachencode MQRC_SELECTOR_INVALID_FOR_TYPE fehl.

Wenn *SelectionString* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der „MQCHARV - Zeichenfolge variabler Länge" auf Seite 277-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_SELECTION_STRING_ERROR fehl. Die maximale Länge von *SelectionString* beträgt MQ_SELECTOR_LENGTH.

Die Verwendung von *SelectionString* (Auswahlzeichenfolge) wird im Abschnitt Selektoren erläutert.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQOD_STRUC_ID
ID für Objektdeskriptorstruktur.

Für die Programmiersprache C ist auch die Konstante MQOD_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQOD_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQOD_STRUC_ID.

UnknownDestCount (MQLONG)

Dies ist die Anzahl der als ferne Warteschlangen aufgelösten und erfolgreich geöffneten Warteschlangen in der Verteilerliste. Wenn es vorhanden ist, wird dieses Feld auch gesetzt, wenn eine einzelne Warteschlange geöffnet wird, die nicht zu einer Verteilerliste gehört.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn *Version* kleiner als MQOD_VERSION_1ist.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Bei dem Wert muss es sich um eine der folgenden Möglichkeiten handeln:

MQOD_VERSION_1
Version-1 Objektdeskriptorstruktur.

MQOD_VERSION_2
Version-2 Objektdeskriptorstruktur.

MQOD_VERSION_3
Version-3 Objektdeskriptorstruktur.

MQOD_VERSION_4
Version-4 Objektdeskriptorstruktur.

Sämtliche Versionen werden in allen WebSphere MQ V7.0-Umgebungen unterstützt.

Felder, die nur in den aktuelleren Versionen der Struktur vorliegen, sind in den Beschreibungen der Felder als solche angegeben. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQOD_CURRENT_VERSION
Aktuelle Version der Objektdeskriptorstruktur.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQOD_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQOD

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQOD_STRUC_ID	'OD¬¬'
Version	MQOD_VERSION_1	1
ObjectType	MQOT_Q	1

Name des Felds	Name der Konstante	Wert der Konstanten
ObjectName	--	Nullzeichenfolge oder Leerzeichen.
ObjectQMgrName	--	Nullzeichenfolge oder Leerzeichen.
DynamicQName	--	'CSQ.*' unter z/OS; andernfalls 'AMQ.*'
AlternateUserId	--	Nullzeichenfolge oder Leerzeichen.
RecsPresent	--	0
KnownDestCount	--	0
UnknownDestCount	--	0
InvalidDestCount	--	0
ObjectRecOffset	--	0
ResponseRecOffset	--	0
ObjectRecPtr	--	Nullzeiger oder Null Byte
ResponseRecPtr	--	Nullzeiger oder Null Byte
AlternateSecurityId	MQSID_NONE	Nullen
ResolvedQName	--	Nullzeichenfolge oder Leerzeichen.
ResolvedQMgrName	--	Nullzeichenfolge oder Leerzeichen.
ObjectString	MQCHARV_DEFAULT	Wie für MQCHARV definiert
SelectionString	MQCHARV_DEFAULT	Wie für MQCHARV definiert
ResObjectString	MQCHARV_DEFAULT	Wie für MQCHARV definiert
ResolvedType	MQOT_NONE	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQOD_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQOD MyOD = {MQOD_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQOD MQOD;
struct tagMQOD {
  MQCHAR4    StrucId;              /* Structure identifier */
  MQLONG     Version;              /* Structure version number */
  MQLONG     ObjectType;           /* Object type */
  MQCHAR48   ObjectName;           /* Object name */
  MQCHAR48   ObjectQMgrName;       /* Object queue manager name */
```

```
  MQCHAR48   DynamicQName;         /* Dynamic queue name */
  MQCHAR12   AlternateUserId;      /* Alternate user identifier */
  /* Ver:1 */
  MQLONG     RecsPresent;          /* Number of object records present */
  MQLONG     KnownDestCount;       /* Number of local queues opened
                                      successfully */
  MQLONG     UnknownDestCount;     /* Number of remote queues opened
                                      successfully */
  MQLONG     InvalidDestCount;     /* Number of queues that failed to
                                      open */
  MQLONG     ObjectRecOffset;      /* Offset of first object record from
                                      start of MQOD */
  MQLONG     ResponseRecOffset;    /* Offset of first response record
                                      from start of MQOD */
  MQPTR      ObjectRecPtr;         /* Address of first object record */
  MQPTR      ResponseRecPtr;       /* Address of first response record */
  /* Ver:2 */
  MQBYTE40   AlternateSecurityId;  /* Alternate security identifier */
  MQCHAR48   ResolvedQName;        /* Resolved queue name */
  MQCHAR48   ResolvedQMgrName;     /* Resolved queue manager name */
  /* Ver:3 */
  MQCHARV    ObjectString;         /* Object Long name */
  MQCHARV    SelectionString;      /* Message Selector */
  MQCHARV    ResObjectString;      /* Resolved Long object name*/
  MQLONG     ResolvedType          /* Alias queue resolved
                                      oject type */
  /* Ver:4 */
};
```

COBOL-Delaration

```
** 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.
```

Deklaration in PL/I

```
dcl
 1 MQOD based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version number */
  3 ObjectType          fixed bin(31), /* Object type */
  3 ObjectName          char(48),      /* Object name */
  3 ObjectQMgrName      char(48),      /* Object queue manager name */
  3 DynamicQName        char(48),      /* Dynamic queue name */
  3 AlternateUserId     char(12),      /* Alternate user identifier */
  3 RecsPresent         fixed bin(31), /* Number of object records
                                          present */
  3 KnownDestCount      fixed bin(31), /* Number of local queues opened
                                          successfully */
  3 UnknownDestCount    fixed bin(31), /* Number of remote queues opened
                                          successfully */
  3 InvalidDestCount    fixed bin(31), /* Number of queues that failed to
                                          open */
  3 ObjectRecOffset     fixed bin(31), /* Offset of first object record
                                          from start of MQOD */
  3 ResponseRecOffset   fixed bin(31), /* Offset of first response record
                                          from start of MQOD */
  3 ObjectRecPtr        pointer,       /* Address of first object record */
  3 ResponseRecPtr      pointer,       /* Address of first response
                                          record */
  3 AlternateSecurityId char(40),      /* Alternate security identifier */
  3 ResolvedQName       char(48),      /* Resolved queue name */
  3 ResolvedQMgrName    char(48),      /* Resolved queue manager name */
  3 ObjectString,                      /* Object Long name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 SelectionString,                   /* Message Selection */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 ResObjectString,                   /* Resolved Long object name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
    3 ResolvedType       fixed bin(31); /* Alias queue resolved object type */
```

Deklaration in High Level Assembler

```
MQOD                            DSECT
MQOD_STRUCID                    DS   CL4   Structure identifier
MQOD_VERSION                    DS   F     Structure version number
MQOD_OBJECTTYPE                 DS   F     Object type
MQOD_OBJECTNAME                 DS   CL48  Object name
MQOD_OBJECTQMGRNAME             DS   CL48  Object queue manager name
MQOD_DYNAMICQNAME               DS   CL48  Dynamic queue name
MQOD_ALTERNATEUSERID            DS   CL12  Alternate user identifier
MQOD_RECSPRESENT                DS   F     Number of object records present
MQOD_KNOWNDESTCOUNT             DS   F     Number of local queues opened
*                                          successfully
MQOD_UNKNOWNDESTCOUNT           DS   F     Number of remote queues opened
*                                          successfully
MQOD_INVALIDDESTCOUNT           DS   F     Number of queues that failed to
*                                          open
MQOD_OBJECTRECOFFSET            DS   F     Offset of first object record from
*                                          start of MQOD
MQOD_RESPONSERECOFFSET          DS   F     Offset of first response record
*                                          from start of MQOD
MQOD_OBJECTRECPTR               DS   F     Address of first object record
MQOD_RESPONSERECPTR             DS   F     Address of first response record
MQOD_ALTERNATESECURITYID        DS   XL40  Alternate security identifier
MQOD_RESOLVEDQNAME              DS   CL48  Resolved queue name
MQOD_RESOLVEDQMGRNAME           DS   CL48  Resolved queue manager name
MQOD_OBJECTSTRING               DS   F     Object Long name
MQOD_OBJECTSTRING_VSPTR         DS   F     Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET      DS   F     Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSIZE     DS   F     size of buffer
MQOD_OBJECTSTRING_VSLENGTH      DS   F     Length of variable length string
MQOD_OBJECTSTRING_VSCCSID       DS   F     CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH        EQU  *- MQOD_OBJECTSTRING
                                ORG  MQOD_OBJECTSTRING
MQOD_OBJECTSTRING_AREA          DS   CL(MQOD_OBJECTSTRING_LENGTH)
*
MQOD_SELECTIONSTRING            DS   F     Message Selector
MQOD_SELECTIONSTRING_VSPTR      DS   F     Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET   DS   F     Offset of variable length string
MQOD_SELECTIONSTRING_VSBUFSIZE  DS   F     size of buffer
MQOD_SELECTIONSTRING_VSLENGTH   DS   F     Length of variable length string
MQOD_SELECTIONSTRING_VSCCSID    DS   F     CCSID of variable length string
MQOD_SELECTIONSTRING_LENGTH     EQU  *- MQOD_SELECTIONSTRING
                                ORG  MQOD_SELECTIONSTRING
MQOD_SELECTIONSTRING_AREA       DS   CL(MQOD_SELECTIONSTRING_LENGTH)
*
MQOD_RESOBJECTSTRING            DS   F     Resolved Long object name
MQOD_RESOBJECTSTRING_VSPTR      DS   F     Address of variable length string
MQOD_RESOBJECTSTRING_VSOFFSET   DS   F     Offset of variable length string
MQOD_RESOBJECTSTRING_VSBUFSIZE  DS   F     size of buffer
MQOD_RESOBJECTSTRING_VSLENGTH   DS   F     Length of variable length string
MQOD_RESOBJECTSTRING_VSCCSID    DS   F     CCSID of variable length string
MQOD_RESOBJECTSTRING_LENGTH     EQU  *- MQOD_RESOBJECTSTRING
                                ORG  MQOD_RESOBJECTSTRING
MQOD_RESOBJECTSTRING_AREA       DS   CL(MQOD_RESOBJECTSTRING_LENGTH)
MQOD_RESOLVEDTYPE               DS   F     Alias queue object resolved type
*
MQOD_LENGTH                     EQU  *-MQOD
                                ORG  MQOD
MQOD_AREA                       DS   CL(MQOD_LENGTH)
```

Deklaration in Visual Basic

```
Type MQOD
  StrucId             As String*4  'Structure identifier'
  Version             As Long      'Structure version number'
  ObjectType          As Long      'Object type'
  ObjectName          As String*48 'Object name'
  ObjectQMgrName      As String*48 'Object queue manager name'
  DynamicQName        As String*48 'Dynamic queue name'
  AlternateUserId     As String*12 'Alternate user identifier'
  RecsPresent         As Long      'Number of object records present'
  KnownDestCount      As Long      'Number of local queues opened'
                                   'successfully'
  UnknownDestCount    As Long      'Number of remote queues opened'
                                   'successfully'
  InvalidDestCount    As Long      'Number of queues that failed to'
                                   'open'
```

```
   ObjectRecOffset     As Long      'Offset of first object record from'
                                    'start of MQOD'
   ResponseRecOffset   As Long      'Offset of first response record'
                                    'from start of MQOD'
   ObjectRecPtr        As MQPTR     'Address of first object record'
   ResponseRecPtr      As MQPTR     'Address of first response record'
   AlternateSecurityId As MQBYTE40  'Alternate security identifier'
   ResolvedQName       As String*48 'Resolved queue name'
   ResolvedQMgrName    As String*48 'Resolved queue manager name'
End Type
```

MQOR - Objektdatensatz

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 521. Felder in MQOR

Feld	Beschreibung	Thema
ObjectName	Objektname	ObjectName
ObjectQMgrName	Name des Objektwarteschlangenmanagers	ObjectQMgrName

Überblick für MQOR

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Verwenden Sie die MQOR-Struktur, um den Namen der Warteschlange und des Warteschlangenmanagers einer einzelnen Zielwarteschlange anzugeben. MQOR ist eine Eingabestruktur für die MQOPEN- und MQPUT1-Aufrufe.

Zeichensatz und Codierung: Die Daten in MQOR müssen dem Zeichensatz entsprechen, der von dem *CodedCharSetId*-Attribut des Warteschlangenmanagers vorgegeben wird, und der Codierung des lokalen Warteschlangenmanagers, die von MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Verwendung: Sie können eine Liste von Warteschlangen öffnen, indem Sie beim MQOPEN-Aufruf eine Feldgruppe dieser Strukturen angeben; diese Liste wird *Verteilerliste* genannt. Jede Nachricht, die unter Verwendung der Warteschlangenkennung eingereiht wird, die der MQOPEN-Aufruf zurückgibt, wird - vorausgesetzt, die entsprechende Warteschlange wurde erfolgreich geöffnet - in jeder der Warteschlangen in der Liste eingereiht.

Felder für MQOR

Die MQMDE-Struktur enthält die folgenden Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

ObjectName (MQCHAR48)

Dies entspricht dem *ObjectName*-Feld in der MQOD-Struktur (weitere Informationen finden Sie im Abschnitt über MQOD), wobei zusätzlich Folgendes zutreffen muss:

- Es muss der Name einer Warteschlange sein.
- Es darf nicht der Name einer Modellwarteschlange sein.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ObjectQMgrName (MQCHAR48)

Dies entspricht dem *ObjectQMgrName*-Feld in der MQOD-Struktur (weitere Informationen finden Sie im Abschnitt über MQOD).

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

Anfangswerte und Sprachendeklarationen für MQOR

Tabelle 522. Anfangswerte von Feldern in MQOR für MQOR

Name des Felds	Name der Konstante	Wert der Konstanten
ObjectName	--	Nullzeichenfolge oder Leerzeichen.
ObjectQMgrName	--	Nullzeichenfolge oder Leerzeichen.

Anmerkungen:

1. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
2. In der Programmiersprache C enthält die MakrovariableMQOR_DEFAULT die oben angegebenen Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQOR MyOR = {MQOR_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
  MQCHAR48  ObjectName;      /* Object name */
  MQCHAR48  ObjectQMgrName;  /* Object queue manager name */
};
```

COBOL-Delaration

```
**   MQOR structure
  10 MQOR.
**    Object name
   15 MQOR-OBJECTNAME     PIC X(48).
**    Object queue manager name
   15 MQOR-OBJECTQMGRNAME PIC X(48).
```

Deklaration in PL/I

```
dcl
 1 MQOR based,
  3 ObjectName     char(48), /* Object name */
  3 ObjectQMgrName char(48); /* Object queue manager name */
```

Deklaration in Visual Basic

```
Type MQOR
  ObjectName     As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

MQPD – Eigenschaftsdeskriptor

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 523. Felder im MQPD		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options
Support	Erforderliche Unterstützung für Nachrichteneigenschaft	Support
Context	Nachrichtenkontext, zu dem die Eigenschaft gehört	Kontext
CopyOptions	Kopieroptionen, zu denen die Eigenschaft gehört	CopyOptions

Überblick für MQPD

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS und WebSphere MQ MQI-Clients.

Zweck: **MQPD** wird zum Definieren der Attribute einer Eigenschaft verwendet. Die Struktur stellt einen Ein-/Ausgabeparameter im MQSETMP-Aufruf und einen Ausgabeparameter im MQINQMP-Aufruf dar.

Zeichensatz und Codierung: Daten im **MQPD** müssen im Zeichensatz der Anwendung und in der Codierung der Anwendung (**MQENC_NATIVE**) vorliegen.

Felder für MQPD

Die MQPD-Struktur enthält die folgenden Felder; die Felder werden in **alphabetischer Reihenfolge** beschrieben:

Context (MQLONG)

Beschreibt, zu welchem Nachrichtenkontext die Eigenschaft gehört.

Wenn ein Warteschlangenmanager eine Nachricht empfängt, die eine von WebSphere MQ definierte Eigenschaft enthält, die der Warteschlangenmanager als fehlerhaft erkennt, dann korrigiert der Warteschlangenmanager den Wert im Feld *Context*.

Die folgende Option kann angegeben werden:

MQPD_USER_CONTEXT
Die Eigenschaft wird dem Benutzerkontext zugeordnet.

Um eine dem Benutzerkontext zugeordnete Eigenschaft über den MQSETMP-Aufruf festzulegen, ist keine besondere Berechtigung erforderlich.

Bei einem Warteschlangenmanager von WebSphere MQ Version 7.0 wird eine Eigenschaft, die dem Benutzerkontext zugeordnet ist, in der für MQOO_SAVE_ALL_CONTEXT beschriebenen Art und Weise gespeichert. Ein MQPUT-Aufruf, für den MQPMO_PASS_ALL_CONTEXT angegeben wurde, veranlasst das Kopieren der Eigenschaft vom gespeicherten Kontext in die neue Nachricht.

Ist die zuvor beschriebene Option nicht erforderlich, kann die folgende Option verwendet werden:

MQPD_NO_CONTEXT
Die Eigenschaft ist keinem Nachrichtenkontext zugeordnet.

Ein nicht erkannter Wert wird mit dem Ursachencode (*Reason*) MQRC_PD_ERROR abgelehnt.

Dies ist ein Ein-/Ausgabefeld im MQSETMP-Aufruf und ein Ausgabefeld im MQINQMP-Aufruf. Der Anfangswert dieses Felds ist MQPD_NO_CONTEXT.

CopyOptions (MQLONG)

Beschreibt, in welchen Nachrichtentyp die Eigenschaft kopiert werden soll. Dies ist ein Nur-Ausgabe-Feld für erkannte WebSphere MQ-definierte Eigenschaften; WebSphere MQ legt den korrekten Wert fest.

Wenn ein Warteschlangenmanager eine Nachricht erhält, die eine mit WebSphere MQ definierte Eigenschaft enthält, die der Warteschlangenmanager als falsch erkennt, korrigiert der Warteschlangenmanager den Wert des Feldes *CopyOptions*.

Sie können eine oder mehrere dieser Optionen angeben; wenn Sie mehr als eine benötigen, können die Werte:

- Hinzufügen (fügen Sie dieselbe Konstante nicht mehrmals hinzu) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

MQCOPY_FORWARD
Diese Eigenschaft wird in eine Nachricht kopiert, die weitergeleitet wird.

MQCOPY_PUBLISH
Diese Eigenschaft wird in die Nachricht kopiert, die beim Veröffentlichen einer Nachricht von einem Subskribenten empfangen wird.

MQCOPY_REPLY
Diese Eigenschaft wird in eine Antwortnachricht kopiert.

MQCOPY_REPORT
Diese Eigenschaft wird in eine Berichtsnachricht kopiert.

MQCOPY_ALL
Diese Eigenschaft wird in alle nachfolgenden Nachrichten kopiert.

Standardoption: Die folgende Option kann zur Bereitstellung der Standardkopieroptionen angegeben werden:

MQCOPY_DEFAULT
Diese Eigenschaft wird in eine weiterzuleitende Nachricht, in eine Berichtsnachricht oder in eine Nachricht kopiert, die von einem Subskribenten empfangen wird, wenn eine Nachricht veröffentlicht wird.

Diese Angabe entspricht der kombinierten Angabe der Optionen MQCOPY_FORWARD, plus MQCOPY_REPORT, plus MQCOPY_PUBLISH.

Verwenden Sie die folgende Option, wenn keine der oben beschriebenen Optionen erforderlich ist:

MQCOPY_NONE
Verwenden Sie diesen Wert, um anzuzeigen, dass keine anderen Kopieroptionen angegeben sind. Auf Programmebene besteht keine Beziehung zwischen dieser Eigenschaft und den nachfolgenden Nachrichten. Dieser Wert wird immer für Nachrichtendeskriptoreigenschaften zurückgegeben.

Dies ist ein Ein-/Ausgabefeld im MQSETMP-Aufruf und ein Ausgabefeld im MQINQMP-Aufruf. Der Anfangswert dieses Felds ist MQCOPY_DEFAULT.

Optionen (MQLONG)

Folgende Werte sind möglich:

MQPD_NONE
Keine Optionen angegeben

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQPD_NONE.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQPD_STRUC_ID
Die ID für die Struktur des Eigenschaftsdeskriptors.

Für die Programmiersprache C wird die Konstante **MQPD_STRUC_ID_ARRAY** ebenfalls definiert. Der Wert stimmt mit dem Wert von **MQPD_STRUC_ID** überein, es handelt sich dabei jedoch um eine Gruppe von Zeichen und nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQPD_STRUC_ID**.

Support (MQLONG)

Dieses Feld beschreibt, welche Unterstützung für die Nachrichteneigenschaft vom Warteschlangenmanager erforderlich ist, damit die Nachricht, die diese Eigenschaft enthält, in eine Warteschlange eingereiht werden kann. Dies gilt nur für WebSphere MQ-definierte Eigenschaften; die Unterstützung für alle anderen Eigenschaften ist optional.

Das Feld wird automatisch auf den korrekten Wert eingestellt, wenn die WebSphere MQ-definierte Eigenschaft dem Warteschlangenmanager bekannt ist. Wenn die Eigenschaft nicht erkannt wird, dann wird MQPD_SUPPORT_OPTIONAL zugeordnet. Wenn ein Warteschlangenmanager eine Nachricht empfängt, die eine von WebSphere MQ definierte Eigenschaft enthält, die der Warteschlangenmanager als fehlerhaft erkennt, dann korrigiert der Warteschlangenmanager den Wert im Feld *Support*.

Wenn eine von WebSphere MQ definierte Eigenschaft mit dem MQSETMP-Aufruf für eine Nachrichtenkennung gesetzt wird, für die die Option MQCMHO_NO_VALIDATION definiert wurde, dann wird *Support* als Eingabefeld festgelegt. Auf diese Weise kann eine Anwendung eine WebSphere MQ-definierte Eigenschaft einreihen, und zwar mit dem richtigen Wert, wenn die Eigenschaft vom verbundenen Warteschlangenmanager nicht unterstützt wird, die Nachricht aber von einem anderen Warteschlangenmanager verarbeitet werden soll.

Der Wert MQPD_SUPPORT_OPTIONAL wird immer Eigenschaften zugeordnet, bei denen es sich nicht um von WebSphere MQ definierte Eigenschaften handelt.

Wenn ein Warteschlangenmanager von WebSphere MQ Version 7.0, der Nachrichteneigenschaften unterstützt, eine Eigenschaft empfängt, die einen nicht erkannten Wert für *Support* enthält, dann wird die Eigenschaft wie folgt behandelt:

- als ob MQPD_SUPPORT_REQUIRED angegeben worden wäre, wenn einer der nicht erkannten Werte in MQPD_REJECT_UNSUP_MASK enthalten ist.
- als ob MQPD_SUPPORT_REQUIRED_IF_LOCAL angegeben worden wäre, wenn einer der nicht erkannten Werte in MQPD_ACCEPT_UNSUP_IF_XMIT_MASK enthalten ist.
- als ob MQPD_SUPPORT_OPTIONAL in anderen Fällen angegeben worden wäre.

Vom MQINQMP-Aufruf wird einer der folgenden Werte zurückgegeben oder einer der Werte kann angegeben werden, wenn der MQSETMP-Aufruf bei einer Nachrichtenkennung verwendet wird, bei der die Option MQCMHO_NO_VALIDATION gesetzt ist:

MQPD_SUPPORT_OPTIONAL
Die Eigenschaft wird von einem Warteschlangenmanager auch dann akzeptiert, wenn sie nicht unterstützt wird. Die Eigenschaft kann gelöscht werden, damit die Nachricht an einen Warteschlangenmanager weitergeleitet werden kann, der keine Nachrichteneigenschaften unterstützt. Dieser Wert wird auch Eigenschaften zugewiesen, die nicht WebSphere MQ-definiert sind.

MQPD_SUPPORT_REQUIRED
Die Unterstützung für die Eigenschaft ist erforderlich. Die Nachricht wird von einem Warteschlangenmanager, der die WebSphere MQ-definierte Eigenschaft nicht unterstützt, zurückgewiesen. Der MQPUT- oder der MQPUT1-Aufruf schlägt mit dem Beendigungscode MQCC_FAILED und dem Ursachencode MQRC_UNSUPPORTED_PROPERTY fehl.

MQPD_SUPPORT_REQUIRED_IF_LOCAL
Die Nachricht wird von einem Warteschlangenmanager, der die WebSphere MQ-definierte Eigenschaft nicht unterstützt, zurückgewiesen, wenn die Nachricht für eine lokale Warteschlange bestimmt ist. Der MQPUT- oder der MQPUT1-Aufruf schlägt mit dem Beendigungscode MQCC_FAILED und dem Ursachencode MQRC_UNSUPPORTED_PROPERTY fehl.

Der MQPUT- oder MQPUT1-Aufruf ist erfolgreich, wenn die Nachricht für einen fernen Warteschlangenmanager bestimmt ist.

Dies ist ein Ausgabefeld im MQINQMP-Aufruf und ein Eingabefeld im MQSETMP-Aufruf, wenn die Nachrichtenkennung mit der Option MQCMHO_NO_VALIDATION erstellt wurde. Der Anfangswert dieses Felds ist MQPD_SUPPORT_OPTIONAL.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQPD_VERSION_1
Version-1 der Struktur des Eigenschaftsdeskriptors.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQPD_CURRENT_VERSION
Aktuelle Version der Struktur des Eigenschaftsdeskriptors.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQPD_VERSION_1**.

Anfangswerte und Sprachendeklarationen für MQPD

Tabelle 524. Anfangswerte der Felder in MQPD

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQPD_STRUC_ID	'PD'
Version	MQPD_VERSION_1	1
Options	MQPD_NONE	0
Support	MQPD_SUPPORT_OPTIONAL	0
Context	MQPD_NO_CONTEXT	0
CopyOptions	MQCOPY_DEFAULT	0

Anmerkungen:

1. In der Programmiersprache C enthält die Makrovariable MQPD_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQPD MyPD = {MQPD_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
  MQCHAR4  StrucId;     /* Structure identifier */
  MQLONG   Version;     /* Structure version number */
  MQLONG   Options;     /* Options that control the action of
                           MQSETMP and MQINQMP */
  MQLONG   Support;     /* Property support option */
  MQLONG   Context;     /* Property context */
  MQLONG   CopyOptions; /* Property copy options */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQPD based,
  3 StrucId     char(4),       /* Structure identifier */
  3 Version     fixed bin(31), /* Structure version number */
  3 Options     fixed bin(31), /* Options that control the action
                                  of MQSETMP and MQINQMP */
  3 Support     fixed bin(31), /* Property support option */
  3 Context     fixed bin(31), /* Property context */
  3 CopyOptions fixed bin(31); /* Property copy options */
```

Deklaration in High Level Assembler

```
MQPD             DSECT
MQPD_STRUCID     DS   CL4    Structure identifier
MQPD_VERSION     DS   F      Structure version number
MQPD_OPTIONS     DS   F      Options that control the
*                            action of MQSETMP and MQINQMP
MQPD_SUPPORT     DS   F      Property support option
MQPD_CONTEXT     DS   F      Property context
MQPD_COPYOPTIONS DS   F      Property copy options
MQPD_LENGTH      EQU  *-MQPD
MQPD_AREA        DS   CL(MQPD_LENGTH)
```

MQPMO - Nachrichteneinreihungsoptionen

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 525. MQPMO-Struktur

Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen zur Steuerung der Aktion von MQPUT und MQPUT1	Options
Timeout	Reserved	Zeitlimit
Context	Objektkennung der Eingabewarteschlange	Kontext
KnownDestCount	Anzahl der Nachrichten, die erfolgreich an lokale Warteschlangen gesendet wurden	KnownDestCount
UnknownDestCount	Anzahl der Nachrichten, die erfolgreich an ferne Warteschlangen gesendet wurden	UnknownDestCount
InvalidDestCount	Anzahl der Nachrichten, die nicht gesendet werden konnten	InvalidDestCount
ResolvedQName	Aufgelöster Name der Zielwarteschlange	ResolvedQName
ResolvedQMgrName	Aufgelöster Name des Zielwarteschlangenmanagers	ResolvedQMgrName
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQPMO_VERSION_2 ist.		
RecsPresent	Anzahl der vorhandenen Datensätze der eingereihten Nachricht oder der vorhandenen Antwortdatensätze	RecsPresent

Tabelle 525. MQPMO-Struktur (Forts.)		
Feld	**Beschreibung**	**Thema**
PutMsgRecFields	Flags, die anzeigen, welche MQPMR-Felder vorhanden sind	PutMsgRecFields
PutMsgRecOffset	Relative Adresse des ersten Datensatzes der eingereihten Nachricht ab dem Anfang des MQPMO	PutMsgRecOffset
ResponseRecOffset	Relative Adresse des ersten Antwortdatensatzes ab dem Anfang des MQPMO	ResponseRecOffset
PutMsgRecPtr	Adresse des ersten Datensatzes der eingereihten Nachricht	PutMsgRecPtr
ResponseRecPtr	Adresse des ersten Antwortdatensatzes	ResponseRecPtr
Anmerkung: Die übrigen Felder werden ignoriert, wenn *Version* kleiner als MQPMO_VERSION_3 ist.		
OriginalMsgHandle	Kennung der ursprünglichen Nachricht	OriginalMsgHandle
NewMsgHandle	Kennung der neuen Nachricht	NewMsgHandle
Action	Typ der ausgeführten Einreihung (PUT) und Beziehung zwischen der ursprünglichen Nachricht, die im Feld *OriginalMsgHandle* angegeben ist, und der neuen Nachricht, die im Feld *NewMsgHandle* angegeben ist	Action
PubLevel	Subskriptionsebene, an die sich die Veröffentlichung richtet	PubLevel

Überblick für MQPMO

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQPMO-Struktur ermöglicht der Anwendung, Optionen anzugeben, die steuern, wie Nachrichten in Warteschlangen eingereiht oder in Themen veröffentlicht werden. Die Struktur ist ein Ein-/Ausgabeparameter in den Aufrufen MQPUT und MQPUT1.

Version: Die aktuelle Version von MQPMO ist MQPMO_VERSION_3. Bestimmte Felder sind nur in bestimmten Versionen von MQPMO verfügbar. Wenn Sie Anwendungen auf mehrere Umgebungen portieren müssen, müssen Sie sicherstellen, dass die Version von MQPMO bei allen Umgebungen konsistent ist. Informationen dazu, welche Felder nur in bestimmten Versionen der Struktur vorliegen, finden Sie im Abschnitt „MQPMO - Nachrichteneinreihungsoptionen” auf Seite 484 und in den Feldbeschreibungen.

Die Headerdateien sowie die COPY- und INCLUDE-Dateien, die für die unterstützten Programmiersprachen bereitgestellt werden, enthalten die aktuellste Version von MQPMO, die von der Umgebung unterstützt wird, wobei der Anfangswert des Feldes *Version* auf MQPMO_VERSION_1 gesetzt ist. Um Felder zu verwenden, die in der Struktur der Version 1 nicht verfügbar sind, muss die Anwendung das Feld *Version* auf die Versionsnummer der erforderlichen Version setzen.

Zeichensatz und Codierung: Die Daten in der MQPMO müssen dem Zeichensatz entsprechen, der von dem *CodedCharSetId*-Attribut des Warteschlangenmanagers vorgegeben wird, und der Codierung des lokalen Warteschlangenmanagers, die von MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQPMO

Die MQMDE-Struktur enthält die folgenden Felder; die Felder sind in **alphabetischer Reihenfolge** beschrieben:

Action (MQLONG)
Dies gibt den Typ der ausgeführten Einreihung (PUT) und die Beziehung zwischen der ursprünglichen Nachricht, die im Feld OriginalMsgHandle angegeben ist, und der neuen Nachricht an, die im Feld NewMsgHandle angegeben ist. Die Eigenschaften der Nachricht werden vom Warteschlangenmanager entsprechend des Werts der angegebenen Aktion ausgewählt.

Sie können sich die Inhalte des Nachrichtendeskriptors mithilfe des Parameters MsgDesc im MQPUT oder über MQPUT1-Aufrufe anzeigen lassen. Alternativ kann auf die Angabe des Parameters MsgDesc verzichtet werden oder man kann angeben, dass dieser nur zur Ausgabe dient, indem MQPMO_MD_FOR_OUTPUT_ONLY im Optionsfeld der MQPMO-Struktur eingefügt wird.

Wird der Parameter MsgDesc nicht bereitgestellt oder ist er als "output-only" definiert, wird der Nachrichtendeskriptor für die neue Nachricht aus den Nachrichtenkennungsfeldern von MQPMO ausgefüllt, entsprechend den in diesem Thema behandelten Regeln.

Die unter Kontextinformationen steuern beschriebene Kontexteinstellung und Übergabeaktivitäten werden wirksam, nachdem der Nachrichtendeskriptor zusammengesetzt wurde.

Bei Angabe eines falschen Aktionswerts schlägt der Aufruf mit dem Ursachencode MQRC_ACTION_ERROR fehl.

Es kann jede der folgenden Optionen angegeben werden:

MQACTP_NEW
Eine neue Nachricht wird eingereiht und das Programm gibt keine Beziehung zu einer früheren Nachricht an. Der Nachrichtendeskriptor setzt sich wie folgt zusammen:

- Wenn im MQPUT- oder MQPUT1-Aufruf eine MsgDesc bereitgestellt wird und in den MQPMO.Options MQPMO_MD_FOR_OUTPUT_ONLY nicht enthalten ist, wird dieser unverändert als Nachrichtendeskriptor verwendet.
- Wird keine MsgDesc angegeben oder die MQPMO.Options enthalten MQPMO_MD_FOR_OUTPUT_ONLY, generiert der Warteschlangenmanager den Nachrichtendeskriptor anhand einer Kombination aus Eigenschaften aus OriginalMsgHandle und NewMsgHandle. Alle in der neuen Nachrichtenkennung explizit festgelegten Nachrichtendeskriptorfelder haben Vorrang vor denen in der ursprünglichen Nachrichtenkennung.

Die Nachrichtendaten werden aus dem Pufferparameter MQPUT oder MQPUT1 übernommen.

MQACTP_FORWARD
Eine zuvor abgerufene Nachricht wird weitergeleitet. Die ursprüngliche Nachrichtenkennung gibt die zuvor abgerufene Nachricht an.

Die neue Nachrichtenkennung gibt alle Änderungen an den Eigenschaften (einschließlich des Nachrichtendeskriptors) im Vergleich zur ursprünglichen Nachrichtenkennung an.

Der Nachrichtendeskriptor setzt sich wie folgt zusammen:

- Wenn im MQPUT- oder MQPUT1-Aufruf eine MsgDesc bereitgestellt wird und in den MQPMO.Options MQPMO_MD_FOR_OUTPUT_ONLY nicht enthalten ist, wird dieser unverändert als Nachrichtendeskriptor verwendet.
- Wird keine MsgDesc angegeben oder die MQPMO.Options enthalten MQPMO_MD_FOR_OUTPUT_ONLY, generiert der Warteschlangenmanager den Nachrichtendeskriptor anhand einer Kombination aus Eigenschaften aus OriginalMsgHandle und NewMsgHandle. Alle in der neuen Nachrichtenkennung explizit festgelegten Nachrichtendeskriptorfelder haben Vorrang vor denen in der ursprünglichen Nachrichtenkennung.
- Wenn in den MQPMO.Options eine MQPMO_NEW_MSG_ID oder MQPMO_NEW_CORREL_ID angegeben ist, wird diese berücksichtigt.

Die Nachrichteneigenschaften setzen sich wie folgt zusammen:

- Alle Eigenschaften aus der ursprünglichen Nachrichtenkennung, die MQCOPY_FORWARD in den MQPD.CopyOptions enthalten
- Alle Eigenschaften aus der neuen Nachrichtenkennung. Für jede Eigenschaft in der neuen Nachrichtenkennung, die denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wird der Wert aus der neuen Nachrichtenkennung übernommen. Die einzige Ausnahme zu dieser Regel ist der Sonderfall, wenn die Eigenschaft in der neuen Nachrichtenkennung denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wobei der Wert der Eigenschaft Null ist. In diesem Fall wird die Eigenschaft aus der Nachricht entfernt.

Die weiterzuleitenden Nachrichtendaten werden aus dem Pufferparameter MQPUT oder MQPUT1 übernommen.

MQACTP_REPLY
Eine zuvor abgerufene Nachricht wird beantwortet. Die ursprüngliche Nachrichtenkennung gibt die zuvor abgerufene Nachricht an.

Die neue Nachrichtenkennung gibt alle Änderungen an den Eigenschaften (einschließlich des Nachrichtendeskriptors) im Vergleich zur ursprünglichen Nachrichtenkennung an.

Der Nachrichtendeskriptor setzt sich wie folgt zusammen:

- Wenn im MQPUT- oder MQPUT1-Aufruf eine MsgDesc bereitgestellt wird und in den MQPMO.Options MQPMO_MD_FOR_OUTPUT_ONLY nicht enthalten ist, wird dieser unverändert als Nachrichtendeskriptor verwendet.
- Wird keine MsgDesc angegeben ist oder die MQPMO.Options enthalten MQPMO_MD_FOR_OUTPUT_ONLY, werden die ursprünglichen Felder des Nachrichtendeskriptors wie folgt ausgewählt:

Tabelle 526. Umsetzung der Antwortnachrichtenkennung

Feld im MQMD	**Verwendeter Wert**
Bericht	Sind MQRO_PASS_DISCARD_AND_EXPIRY und MQRO_DISCARD_MSG gesetzt: MQRO_DISCARD_MSG andernfalls MQRO_NONE
MsgType	MQMT_REPLY
Verfall	Sind MQRO_PASS_DISCARD_AND_EXPIRY gesetzt: Kopiert aus Eingabenachricht andernfalls MQEI_UNLIMITED
Feedback	MQFB_NONE
MsgId	Ist MQPMO_NEW_MSG_ID gesetzt: Eine neue Nachrichten-ID wird generiert ansonsten, falls MQRO_PASS_MSG_ID gesetzt ist: Kopiert aus Eingabenachricht andernfalls MQMI_NONE

Tabelle 526. Umsetzung der Antwortnachrichtenkennung (Forts.)	
Feld im MQMD	**Verwendeter Wert**
CorrelId	Ist MQPMO_NEW_CORREL_ID gesetzt: Eine neue Korrelations-ID wird generiert ansonsten, falls MQRO_COPY_MSG_ID_TO_COR-REL_ID gesetzt ist: Kopiert aus dem MsgId-Feld der Eingabenachricht ansonsten, falls MQRO_PASS_CORREL_ID gesetzt ist: Kopiert aus dem CorrelId-Feld der Eingabenachricht andernfalls MQCI_NONE
BackoutCount	0
ReplyToQ	Leerzeichen
ReplyToQMgr	Leerzeichen
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- Danach wird der Nachrichtendeskriptor vom neuen Nachrichtenhandle geändert – alle explizit als Eigenschaften im neuen Nachrichtenhandle eingerichteten Nachrichtendeskriptorfelder haben Vorrang vor den Nachrichtendeskriptorfeldern, wie oben beschrieben.

Die Nachrichteneigenschaften setzen sich wie folgt zusammen:

- Alle Eigenschaften aus der ursprünglichen Nachrichtenkennung, die MQCOPY_REPLY in den MQPD.CopyOptions enthalten
- Alle Eigenschaften aus der neuen Nachrichtenkennung. Für jede Eigenschaft in der neuen Nachrichtenkennung, die denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wird der Wert aus der neuen Nachrichtenkennung übernommen. Die einzige Ausnahme zu dieser Regel ist der Sonderfall, wenn die Eigenschaft in der neuen Nachrichtenkennung denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wobei der Wert der Eigenschaft Null ist. In diesem Fall wird die Eigenschaft aus der Nachricht entfernt.

Die weiterzuleitenden Nachrichtendaten werden aus dem Pufferparameter MQPUT/MQPUT1 übernommen.

MQACTP_REPORT

Als Ergebnis einer zuvor abgerufenen Nachricht wird ein Bericht generiert. Die ursprüngliche Nachrichtenkennung gibt die Nachricht an, die das Generieren des Berichts zur Folge hat.

Die neue Nachrichtenkennung gibt alle Änderungen an den Eigenschaften (einschließlich des Nachrichtendeskriptors) im Vergleich zur ursprünglichen Nachrichtenkennung an.

Der Nachrichtendeskriptor setzt sich wie folgt zusammen:

- Wenn im MQPUT- oder MQPUT1-Aufruf eine MsgDesc bereitgestellt wird und in den MQPMO.Options MQPMO_MD_FOR_OUTPUT_ONLY nicht enthalten ist, wird dieser unverändert als Nachrichtendeskriptor verwendet.
- Wird keine MsgDesc angegeben ist oder die MQPMO.Options enthalten MQPMO_MD_FOR_OUTPUT_ONLY, werden die ursprünglichen Felder des Nachrichtendeskriptors wie folgt ausgewählt:

Tabelle 527. Umsetzung der Kennung der Berichtsnachricht	
Feld im MQMD	**Verwendeter Wert**
Bericht	Sind MQRO_PASS_DISCARD_AND_EXPIRY und MQRO_DISCARD_MSG gesetzt: MQRO_DISCARD_MSG andernfalls MQRO_NONE
MsgType	MQMT_REPORT
Verfall	Sind MQRO_PASS_DISCARD_AND_EXPIRY gesetzt: Kopiert aus Eingabenachricht andernfalls MQEI_UNLIMITED
MsgId	Ist MQPMO_NEW_MSG_ID gesetzt: Eine neue Nachrichten-ID wird generiert ansonsten, falls MQRO_PASS_MSG_ID gesetzt ist: Kopiert aus Eingabenachricht andernfalls MQMI_NONE
CorrelId	Ist MQPMO_NEW_CORREL_ID gesetzt: Eine neue Korrelations-ID wird generiert ansonsten, falls MQRO_COPY_MSG_ID_TO_COR-REL_ID gesetzt ist: Kopiert aus dem MsgId-Feld der Eingabenachricht ansonsten, falls MQRO_PASS_CORREL_ID gesetzt ist: Kopiert aus dem CorrelId-Feld der Eingabenachricht andernfalls MQCI_NONE
BackoutCount	0
ReplyToQ	Leerzeichen
ReplyToQMgr	Leerzeichen
OriginalLength	Auf die *BufferLength* gesetzt

- Danach wird der Nachrichtendeskriptor vom neuen Nachrichtenhandle geändert – alle explizit als Eigenschaften im neuen Nachrichtenhandle eingerichteten Nachrichtendeskriptorfelder haben Vorrang vor den Nachrichtendeskriptorfeldern, wie oben beschrieben.

Die Nachrichteneigenschaften setzen sich wie folgt zusammen:

- Alle Eigenschaften aus dem ursprünglichen Bericht, die MQCOPY_REPORT in den MQPD.CopyOptions enthalten
- Alle Eigenschaften aus der neuen Nachrichtenkennung. Für jede Eigenschaft in der neuen Nachrichtenkennung, die denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wird der Wert aus der neuen Nachrichtenkennung übernommen. Die einzige Ausnahme zu dieser Regel ist der Sonderfall, wenn die Eigenschaft in der neuen Nachrichtenkennung denselben Namen hat wie eine Eigenschaft in der ursprünglichen Nachrichtenkennung, wobei der Wert der Eigenschaft Null ist. In diesem Fall wird die Eigenschaft aus der Nachricht entfernt.

Das Feedback-Feld im resultierenden MQMD stellt den zu generierenden Bericht dar. Beim Feedback-Wert MQFB_NONE schlägt der MQPUT- oder MQPUT1-Aufruf mit dem Ursachencode MQRC_FEEDBACK_ERROR fehl.

Zur Auswahl der Benutzerdaten der Berichtsnachricht konsultiert WebSphere MQ die Felder Report und Feedback im resultierenden MQMD sowie die Parameter Buffer und BufferLength des MQPUT- oder MQPUT1-Aufrufs.

- Wird MQFB_COA, MQFB_COD oder MQFB_EXPIRATION zurückgemeldet, wird der Wert des Berichts überprüft.
- Liegt einer der folgenden Fälle vor, werden die vollständigen Nachrichtendaten aus "Buffer" der Länge BufferLength verwendet.
  - Es wird MQFB_EXPIRATION zurückgemeldet und der Bericht enthält MQRO_EXPIRATION_WITH_FULL_DATA
  - Es wird MQFB_COD zurückgemeldet und der Bericht enthält MQRO_COD_WITH_FULL_DATA
  - Es wird MQFB_COA zurückgemeldet und der Bericht enthält MQRO_COA_WITH_FULL_DATA
- Liegt einer der folgenden Fälle vor, werden die ersten 100 Byte der Nachricht (oder BufferLength bei weniger als 100) aus dem Puffer verwendet
  - Es wird MQFB_EXPIRATION zurückgemeldet und der Bericht enthält MQRO_EXPIRATION_WITH_DATA
  - Es wird MQFB_COD zurückgemeldet und der Bericht enthält MQRO_COD_WITH_DATA
  - Es wird MQFB_COA zurückgemeldet und der Bericht enthält MQRO_COA_WITH_DATA
- Wird MQFB_EXPIRATION, MQFB_COD oder MQFB_COA zurückgemeldet und der Bericht enthält nicht die relevanten Optionen *_WITH_FULL_DATA oder *_WITH_DATA für diesen Feedback-Wert, werden keine Benutzerdaten in die Nachricht aufgenommen.
- Wird ein anderer Wert als die oben angegebenen zurückgemeldet, werden normal Buffer und BufferLength verwendet.

In der folgenden Tabelle ist die Ableitung der Benutzerdaten aufgeführt:

Tabelle 528. Quelle der Benutzerdaten

	MQFB_COA	**MQFB_COD**	**MQFB_EXPIRATION**
MQRO_EXPIRATION_WITH_FULL_DATA	--	--	Buffer(Bufferlength)
MQRO_COD_WITH_FULL_DATA	--	Buffer(Bufferlength)	--
MQRO_COA_MIT_FULL_DATA	Buffer(Bufferlength)	--	--
MQRO_EXPIRATION_WITH_DATA	--	--	Buffer(erste 100 Byte)
MQRO_COD_WITH_DATA	--	Buffer(erste 100 Byte)	--
MQRO_COA_WITH_DATA	Buffer(erste 100 Byte)	--	--

Context (MQHOBJ)

Wenn MQPMO_PASS_IDENTITY_CONTEXT oder MQPMO_PASS_ALL_CONTEXT angegeben ist, muss dieses Feld die Kennung der Eingabewarteschlange enthalten, der die der einzureihenden Nachricht zugehörigen Kontextinformation entnommen wird.

Wenn weder MQPMO_PASS_IDENTITY_CONTEXT noch MQPMO_PASS_ALL_CONTEXT angegeben sind, wird diese Feld ignoriert.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist `0`.

InvalidDestCount (MQLONG)

Dies ist die Anzahl der Nachrichten, die nicht an Warteschlangen in der Verteilerliste gesendet werden konnten. Dabei werden Warteschlangen, die nicht geöffnet werden konnten, sowie Warteschlangen, die zwar geöffnet werden konnten, bei denen aber die Put-Operation fehlschlug, berücksichtigt. Dieses Feld wird auch gesetzt, wenn eine Nachricht in einer einzelnen Warteschlange eingereiht wird, die keiner Verteilerliste angehört.

Anmerkung: Dieses Feld wird gesetzt, wenn der *CompCode*-Parameter beim MQPUT- oder MQPUT1-Aufruf MQCC_OK oder MQCC_WARNING ist. Es wird eventuell gesetzt, wenn der *CompCode*-Parameter MQCC_FAILED ist, aber verlassen Sie sich nicht auf diesen Anwendungscode.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld ist nicht festgelegt, wenn die *Version* niedriger ist als MQPMO_VERSION_1.

Dieses Feld ist bei z/OS nicht definiert, da keine Verteilerlisten unterstützt werden.

KnownDestCount (MQLONG)

Dies ist die Anzahl der Nachrichten, welche mit dem aktuellen MQPUT- oder MQPUT1-Aufruf erfolgreich an Warteschlangen in der Verteilerliste gesendet werden konnten, die lokale Warteschlangen sind. Nicht in dieser Anzahl enthalten sind Nachrichten, die an als ferne Warteschlangen aufgelöste Warteschlangen gesandt wurden (auch wenn zur Abspeicherung der Nachricht zunächst eine lokale Übertragungswarteschlange verwendet wird). Dieses Feld wird auch gesetzt, wenn eine Nachricht in einer einzelnen Warteschlange eingereiht wird, die keiner Verteilerliste angehört.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld ist nicht festgelegt, wenn die *Version* niedriger ist als MQPMO_VERSION_1.

Dieses Feld ist bei z/OS nicht definiert, da keine Verteilerlisten unterstützt werden.

NewMsgHandle (MQHMSG)

Dies ist eine optionale Kennung für die eingereihte Nachricht, abhängig vom Wert des Felds "Action". Sie legt die Eigenschaften der Nachricht fest und setzt gegebenenfalls gesetzte Werte von *OriginalMsgHandle* außer Kraft.

Bei Rückgabe vom **MQPUT**- oder **MQPUT1**-Aufruf geben die Inhalte der Kennung an, welche Nachricht eingereiht wurde.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist **MQHM_NONE**. Dieses Feld wird ignoriert, wenn Version kleiner als **MQPMO_VERSION_3** ist.

MQPMO-Optionen (MQLONG)
Das Options-Feld steuert die Ausführung von **MQPUT**- und **MQPUT1**-Aufrufen.

Geltungsbereichsoption. Sie können jede oder keine der MQPMO-Optionen angeben. Wenn mehrere Optionen erforderlich sind, können die Werte, die Sie für die Optionen angeben, auf folgende Arten verwendet werden:

- Die Werte können hinzugefügt werden. Fügen Sie dieselbe Konstante nur einmal hinzu.
- Die Werte können mithilfe der bitweisen ODER-Operation kombiniert werden, falls die Programmiersprache bitweise Operationen unterstützt.

Auf ungültige Kombinationen wird hingewiesen; alle anderen Kombinationen sind gültig.

Die folgende Option steuert den Geltungsbereich der gesendeten Veröffentlichungen:

MQPMO_SCOPE_QMGR

Die Veröffentlichung wird nur an Subskribenten gesendet, die eine Subskription für diesen Warteschlangenmanager eingerichtet haben. Die Veröffentlichung wird nicht an ferne Publish/Subscribe-Warteschlangenmanager weitergeleitet, die eine Subskription für diesen Warteschlangenmanager eingerichtet haben, wodurch das gesamte Verhalten, das mit dem Themenattribut PUBSCOPE festgelegt ist, außer Kraft gesetzt wird.

Anmerkung: Ist diese Option nicht angegeben, wird der Geltungsbereich der Veröffentlichung durch das Themenattribut PUBSCOPE festgelegt.

Veröffentlichungsoptionen. Die folgenden Optionen steuern die Art und Weise, wie Nachrichten zu einem Thema veröffentlicht werden:

MQPMO_SUPPRESS_REPLYTO

Alle Informationen, die in den Feldern *ReplyToQ* und *ReplyToQMgr* des MQMD dieser Veröffentlichung angegeben sind, werden nicht an Abonnenten weitergegeben. Wenn diese Option mit einer Berichtsoption verwendet wird, die *ReplyToQ* erfordert, schlägt der Aufruf mit MQRC_MISSING_REPLY_TO_Q fehl.

MQPMO_RETAIN

Die gesendete Veröffentlichung muss vom Warteschlangenmanager als ständige Veröffentlichung bereitgestellt werden. Bei einer ständigen Veröffentlichung kann ein Subskribent auch noch nach dem Zeitpunkt der Veröffentlichung mit dem Aufruf MQSUBRQ eine Kopie der Veröffentlichung anfordern. Dadurch ist es auch möglich, eine Veröffentlichung an Anwendungen zu senden, die ihre Subskription erst nach dem Zeitpunkt der Veröffentlichung einrichten (sofern dies nicht durch Angabe der Option MQSO_NEW_PUBLICATIONS_ONLY ausgeschlossen wird). Wenn eine Anwendung eine ständige Veröffentlichung erhält, wird durch die Nachrichteneigenschaft MQIsRetained der betreffenden Veröffentlichung darauf hingewiesen.

Es kann auf jedem Knoten der Themenstruktur nur eine ständige Veröffentlichung geben. Daher wird eine bereits vorhandene ständige Veröffentlichung, die durch eine andere Anwendung erfolgte, durch die neue Veröffentlichung ersetzt. Es sollte deshalb vermieden werden, dass mehrere Veröffentlichungskomponenten für dasselbe Thema Nachrichten als ständige Veröffentlichung senden.

Wenn ein Subskribent ständige Veröffentlichungen anfordert, kann die Subskription ein Platzhalterzeichen im Thema enthalten. In diesem Fall kann es eine Übereinstimmung mit mehreren ständigen Veröffentlichungen (auf verschiedenen Knoten in der Themenstruktur) geben, sodass mehrere Veröffentlichungen an die anfordernde Anwendung gesendet werden. Weitere Informationen finden Sie in der Beschreibung des Aufrufs „MQSUBRQ - Subskriptionsanforderung” auf Seite 794.

Informationen zur Interaktion zwischen ständigen Veröffentlichungen und Subskriptionsebenen finden Sie im Abschnitt Veröffentlichungen abfangen.

Wenn diese Option angegeben, aber eine ständige Veröffentlichung nicht möglich ist, wird die betreffende Nachricht nicht veröffentlicht und der Aufruf schlägt mit MQRC_PUT_NOT_RETAINED fehl.

MQPMO_NOT_OWN_SUBS

Über diese Option teilt eine Anwendung dem Warteschlangenmanager mit, dass keine ihrer Veröffentlichungen an Subskriptionen gesendet werden sollen, die Eigentum der Anwendung sind. Subskriptionen gelten als Eigentum derselben Anwendung, wenn die Verbindungskennungen identisch sind.

MQPMO_WARN_IF_NO_SUBS_MATCHED

Wenn keine Subskription mit der Veröffentlichung übereinstimmt, wird der Beendigungscode (*CompCode*) MQCC_WARNING und der Ursachencode MQRC_NO_SUBS_MATCHED zurückgegeben.

Wenn die PUT-Operation MQRC_NO_SUBS_MATCHED zurückgibt, wurde die Veröffentlichung an keine Subskription übergeben. Wenn für die PUT-Operation jedoch die Option MQPMO_RETAIN angegeben ist, wird die Nachricht als ständige Veröffentlichung bereitgestellt und an jede später definierte, übereinstimmende Subskription übergeben.

Eine Subskription für das Thema stimmt mit der Veröffentlichung überein, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Die Nachricht wird an die Subskriptionswarteschlange übergeben.
- Die Nachricht sollte an die Subskriptionswarteschlange übergeben werden, aber ein Problem mit der Warteschlange hat zur Folge, dass die Nachricht nicht in die Warteschlange eingereiht werden kann und deshalb in die Warteschlange für nicht zustellbare Nachrichten gestellt oder gelöscht wurde.
- Es ist ein Weiterleitungsexit definiert, der die Zustellung der Nachricht an die Subskription verhindert.

Eine Subskription für das Thema stimmt nicht mit der Veröffentlichung überein, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Die Subskription enthält eine Auswahlzeichenfolge, die nicht mit der Veröffentlichung übereinstimmt.
- In der Subskription ist die Option MQSO_PUBLICATION_ON_REQUEST angegeben.
- Die Veröffentlichung wird nicht übergeben, weil in der PUT-Operation die Option MQPMO_NOT_OWN_SUBS angegeben wurde und die Subskription mit der ID der Veröffentlichungskomponente übereinstimmt.

Synchronisationspunktoptionen. Die folgenden Optionen beziehen sich auf die Verwendung des MQPUT- oder MQPUT1-Aufrufs in einer Arbeitseinheit:

MQPMO_SYNCPOINT
Die Anforderung wird innerhalb der normalen Arbeitseinheitenprotokolle ausgeführt. Die Nachricht wird erst außerhalb der Arbeitseinheit sichtbar, wenn die Arbeitseinheit festgeschrieben wird. Wird die Arbeitseinheit zurückgesetzt, wird die Nachricht gelöscht.

Wenn MQPMO_SYNCPOINT und MQPMO_NO_SYNCPOINT nicht angegeben sind, wird die Einbeziehung der PUT-Anforderung in Arbeitseinheitenprotokolle durch die Umgebung, in der der Warteschlangenmanager aktiv ist, und nicht durch die Umgebung, in der die Anwendung aktiv ist, bestimmt. Unter z/OSbefindet sich die PUT-Anforderung innerhalb einer Arbeitseinheit. In allen anderen Umgebungen befindet sich die PUT-Anforderung nicht innerhalb einer Arbeitseinheit.

Aufgrund dieses Unterschieds darf eine Anwendung, die Sie portieren möchten, diese Option nicht als Standardeinstellung zulassen; geben Sie explizit entweder MQPMO_SYNCPOINT oder MQPMO_NO_SYNCPOINT an.

Geben Sie nicht MQPMO_SYNCPOINT zusammen mit MQPMO_NO_SYNCPOINT an.

MQPMO_NO_SYNCPOINT
Die Anforderung soll außerhalb der normalen Arbeitseinheitenprotokolle ausgeführt werden. Die Nachricht ist unverzüglich verfügbar und kann nicht durch Zurücksetzen einer Arbeitseinheit gelöscht werden.

Wenn MQPMO_NO_SYNCPOINT und MQPMO_SYNCPOINT nicht angegeben sind, wird die Einbeziehung der PUT-Anforderung in Arbeitseinheitenprotokolle durch die Umgebung, in der der Warteschlangenmanager aktiv ist, und nicht durch die Umgebung, in der die Anwendung aktiv ist, bestimmt. Unter z/OSbefindet sich die PUT-Anforderung innerhalb einer Arbeitseinheit. In allen anderen Umgebungen befindet sich die PUT-Anforderung nicht innerhalb einer Arbeitseinheit.

Aufgrund dieses Unterschieds darf eine Anwendung, die Sie portieren möchten, diese Option nicht als Standardeinstellung zulassen; geben Sie explizit entweder MQPMO_SYNCPOINT oder MQPMO_NO_SYNCPOINT an.

Geben Sie nicht MQPMO_NO_SYNCPOINT zusammen mit MQPMO_SYNCPOINT an.

Nachrichten-ID- und Korrelations-ID-Optionen. Die folgenden Optionen fordern den Warteschlangenmanager auf, eine neue Nachrichten-ID oder Korrelations-ID zu generieren:

MQPMO_NEW_MSG_ID
Der Warteschlangenmanager ersetzt den Inhalt des Feldes *MsgId* im MQMD durch eine neue Nachrichten-ID. Diese Nachrichten-ID wird mit der Nachricht gesendet und bei der Ausgabe mit dem Aufruf MQPUT oder MQPUT1 an die Anwendung zurückgegeben.

Die Option MQPMO_NEW_MSG_ID kann auch angegeben werden, wenn die Nachricht in eine Verteilerliste eingereiht wird. Weitere Informationen finden Sie in der Beschreibung des Felds *MsgId* in der MQPMR-Struktur.

Bei Verwendung dieser Option muss die Anwendung das Feld *MsgId* vor jedem MQPUT- oder MQPUT1-Aufruf nicht mehr auf MQMI_NONE zurücksetzen.

MQPMO_NEW_CORREL_ID
Der Warteschlangenmanager ersetzt den Inhalt des Feldes *CorrelId* im MQMD durch eine neue Korrelations-ID. Diese Korrelations-ID wird mit der Nachricht gesendet und bei der Ausgabe mit dem Aufruf MQPUT oder MQPUT1 an die Anwendung zurückgegeben.

Die Option MQPMO_NEW_CORREL_ID kann ebenfalls angegeben werden, wenn die Nachricht in eine Verteilerliste eingereiht wird. Details finden Sie in der Beschreibung des Felds *CorrelId* in der MQPMR-Struktur.

Die Option MQPMO_NEW_CORREL_ID ist in Situationen hilfreich, in denen die Anwendung eine eindeutige Korrelations-ID erfordert.

Gruppen- und Segmentoptionen. Die folgenden Optionen beziehen sich auf die Verarbeitung von Nachrichten in Gruppen und Segmenten von logischen Nachrichten. Lesen Sie zum besseren Verständnis der Option die nachfolgenden Definitionen.

Achtung: Sie können segmentierte oder gruppierte Nachricht mit Publish/Subscribe nicht verwenden.

Physische Nachricht
Dies ist die kleinste Informationseinheit, die in eine Warteschlange gestellt oder aus einer Warteschlange entfernt werden kann. Sie entspricht häufig der Information, die in einem einzelnen MQPUT-, MQPUT1- oder MQGET-Aufruf angegeben oder abgerufen wird. Jede physische Nachricht besitzt einen eigenen Nachrichtendeskriptor (MQMD). Im Allgemeinen unterscheiden sich physische Nachrichten durch unterschiedliche Werte für die Nachrichten-ID (Feld *MsgId* in MQMD), obwohl dies nicht vom Warteschlangenmanager erzwungen wird.

Logische Nachricht
Eine logische Nachricht ist eine einzelne Einheit mit Anwendungsinformationen nur für Nicht-z/OS -Plattformen. Bleiben Systembedingungen unberücksichtigt, ist eine logische Nachricht dasselbe wie eine physische Nachricht. Wenn logische Nachrichten jedoch sehr groß sind, kann es aufgrund von Systembedingungen ratsam oder nötig sein, eine logische Nachricht in zwei oder mehr physische Nachrichten, sogenannte *Segmente*, aufzuteilen.

Eine logische Nachricht, die segmentiert wurde, besteht aus zwei oder mehr physischen Nachrichten mit derselben Gruppen-ID ungleich null (Feld *GroupId* im MQMD) und derselben Nachrichtenfolgenummer (Feld *MsgSeqNumber* im MQMD). Die Segmente unterscheiden sich durch unterschiedliche Werte für den Segmentoffset (Feld *Offset* in MQMD), der den Offset der Daten in der physischen Nachricht vom Anfang der Daten in der logischen Nachricht angibt. Da jedes Segment eine physische Nachricht ist, haben die Segmente in einer logischen Nachricht normalerweise unterschiedliche Nachrichten-IDs.

Eine logische Nachricht, die nicht in Segmente aufgeteilt wurde, aber für die die sendende Anwendung eine Segmentierung zugelassen hat, besitzt ebenfalls eine Gruppen-ID ungleich null, obwohl es in diesem Fall nur eine einzige physische Nachricht mit dieser Gruppen-ID gibt, wenn die logische Nachricht nicht zu einer Nachrichtengruppe gehört. Logische Nachrichten, für die die sendende Anwendung eine Segmentierung verboten hat, besitzen eine Gruppen-ID null (MQGI_NONE), außer wenn die logische Nachricht zu einer Nachrichtengruppe gehört.

Nachrichtengruppe
Eine Nachrichtengruppe ist eine Gruppe aus einer oder mehreren logischen Nachrichten, die dieselbe Gruppen-ID ungleich null besitzen. Die logischen Nachrichten in der Gruppe unterscheiden sich durch verschiedene Werte für die Nachrichtenfolgenummer, bei der es sich um eine ganze Zahl im Bereich 1 bis *n* handelt, wobei *n* der Anzahl der logischen Nachrichten in der Gruppe entspricht. Wenn eine oder mehrere logische Nachrichten segmentiert sind, enthält die Gruppe mehr als *n* physische Nachrichten.

MQPMO_LOGICAL_ORDER
Diese Option teilt dem Warteschlangenmanager mit, wie die Anwendung Nachrichten in Gruppen und Segmente von logischen Nachrichten einfügt. Sie kann nur für den Aufruf MQPUT angegeben werden; für den Aufruf MQPUT1 ist sie nicht gültig.

Mit MQPMO_LOGICAL_ORDER wird angegeben, dass die Anwendung aufeinanderfolgende MQPUT-Aufrufe für Folgendes verwendet:

1. Segmente werden in jede logische Nachricht nach Systemoffset in aufsteigender Reihenfolge (beginnend bei 0 und ohne Lücken) eingefügt.
2. Alle Segmente werden zunächst vollständig in eine logische Nachricht eingefügt, bevor sie in die nächste logische Nachricht eingefügt werden.
3. Die logischen Nachrichten werden nach Nachrichtenfolgenummer in aufsteigender Reihenfolge (beginnend bei 1 und ohne Lücken) in die einzelnen Nachrichtengruppen eingefügt.IBM WebSphere MQ erhöht die Nachrichtenfolgenummer automatisch.
4. Alle logischen Nachrichten werden zunächst vollständig in eine Nachrichtengruppe eingefügt, bevor sie in die nächste Nachrichtengruppe eingefügt werden.

Ausführliche Informationen zu MQPMO_LOGICAL_ORDER finden Sie im Abschnitt Logische und physische Anordnung

Kontextoptionen. Die folgenden Optionen steuern die Verarbeitung des Nachrichtenkontexts:

MQPMO_NO_CONTEXT
Sowohl Identität als auch Ursprungskontext zeigen durch ihre Einstellung an, dass kein Kontext vorhanden ist. Dies bedeutet, dass die Kontextfelder im MQMD folgende Werte enthalten:

- Leerzeichen in Zeichenfeldern
- Nullzeichen in Bytefeldern
- Nullen in numerischen Feldern

MQPMO_DEFAULT_CONTEXT
Der Nachricht müssen Standardkontextinformationen sowohl für Identität als auch für Ursprung zugeordnet sein. Der Warteschlangenmanager legt die Kontextfelder im Nachrichtendeskriptor wie folgt fest:

Feld im MQMD	Verwendeter Wert
UserIdentifier	Wird, wenn möglich, durch die Umgebung bestimmt; wird andernfalls auf Leerzeichen gesetzt.
AccountingToken	Wird, wenn möglich, durch die Umgebung bestimmt; wird andernfalls auf MQACT_NONE gesetzt.
ApplIdentityData	Wird auf Leerzeichen gesetzt.
PutApplType	Wird durch die Umgebung bestimmt.
PutApplName	Wird, wenn möglich, durch die Umgebung bestimmt; wird andernfalls auf Leerzeichen gesetzt.
PutDate	Wird auf das Datum gesetzt, an dem die Nachricht eingereiht wird.
PutTime	Wird auf die Uhrzeit gesetzt, zu der die Nachricht eingereiht wird.
ApplOriginData	Wird auf Leerzeichen gesetzt.

Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Dies sind die Standardwerte und Aktionen, wenn keine Kontextoptionen angegeben sind.

MQPMO_PASS_IDENTITY_CONTEXT
Der Nachricht müssen Kontextinformationen zugeordnet sein. Der Identitätskontext wird der Warteschlangenkennung entnommen, die im Feld *Context* angegeben ist. Ursprungskontextinformationen werden vom Warteschlangenmanager auf dieselbe Weise wie für MQPMO_DEFAULT_CONTEXT gene-

riert (Werte siehe vorhergehende Tabelle). Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Für den MQPUT-Aufruf muss die Warteschlange mit der Option MQOO_PASS_IDENTITY_CONTEXT (oder einer Option, die sie einschließt) geöffnet worden sein. Für den MQPUT1-Aufruf wird dieselbe Berechtigungsprüfung wie für den MQOPEN-Aufruf mit der Option MQOO_PASS_IDENTITY_CONTEXT durchgeführt.

MQPMO_PASS_ALL_CONTEXT
Der Nachricht müssen Kontextinformationen zugeordnet sein. Der Kontext wird der Warteschlangenkennung entnommen, die im Feld *Context* angegeben ist. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Steuerung von Kontextinformationen.

Für den MQPUT-Aufruf muss die Warteschlange mit der Option MQOO_PASS_ALL_CONTEXT (oder einer Option, die sie einschließt) geöffnet worden sein. Für den MQPUT1-Aufruf wird dieselbe Berechtigungsprüfung wie für den MQOPEN-Aufruf mit der Option MQOO_PASS_ALL_CONTEXT durchgeführt.

MQPMO_SET_IDENTITY_CONTEXT
Der Nachricht müssen Kontextinformationen zugeordnet sein. Die Anwendung gibt den Identitätskontext in der MQMD-Struktur an. Ursprungskontextinformationen werden vom Warteschlangenmanager auf dieselbe Weise wie für MQPMO_DEFAULT_CONTEXT generiert (Werte siehe vorhergehende Tabelle). Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Für den MQPUT-Aufruf muss die Warteschlange mit der Option MQOO_SET_IDENTITY_CONTEXT (oder einer Option, die sie einschließt) geöffnet worden sein. Für den MQPUT1-Aufruf wird dieselbe Berechtigungsprüfung wie für den MQOPEN-Aufruf mit der Option MQOO_SET_IDENTITY_CONTEXT durchgeführt.

MQPMO_SET_ALL_CONTEXT
Der Nachricht müssen Kontextinformationen zugeordnet sein. Die Anwendung gibt den Identitäts-, Ursprungs- und Benutzerkontext in der MQMD-Struktur an. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.

Für den MQPUT-Aufruf muss die Warteschlange mit der Option MQOO_SET_ALL_CONTEXT geöffnet worden sein. Für den MQPUT1-Aufruf wird dieselbe Berechtigungsprüfung wie für den MQOPEN-Aufruf mit der Option MQOO_SET_ALL_CONTEXT durchgeführt.

Es kann nur eine der MQPMO_*_CONTEXT-Kontextoptionen angegeben werden. Wenn Sie keine angeben, wird MQPMO_DEFAULT_CONTEXT vorausgesetzt.

Eigenschaftsoptionen. Die folgende Option bezieht sich auf die Eigenschaften der Nachricht:

MQPMO_MD_FOR_OUTPUT_ONLY
Der Nachrichtendeskriptorparameter darf nur zur Ausgabe verwendet werden, um den Nachrichtendeskriptor der eingereihten Nachricht zurückzugeben. Zur Eingabe müssen die Nachrichtendeskriptorfelder, die dem Feld *NewMsgHandle* oder *OriginalMsgHandle* (oder beiden) in der **MQPMO**-Struktur zugeordnet sind, verwendet werden.

Wenn keine gültige Nachrichtenkennung angegeben ist, schlägt der Aufruf mit Ursachencode **MQRC_MD_ERROR** fehl.

PUT-Antwortoptionen. Die folgenden Optionen steuern die Antwort, die an einen MQPUT- oder MQPUT1-Aufruf zurückgegeben wird. Es kann nur eine dieser Optionen angegeben werden. Wenn MQPMO_ASYNC_RESPONSE und MQPMO_SYNC_RESPONSE nicht angegeben sind, wird MQPMO_RESPONSE_AS_Q_DEF oder MQPMO_RESPONSE_AS_TOPIC_DEF vorausgesetzt.

MQPMO_ASYNC_RESPONSE
Die Option MQPMO_ASYNC_RESPONSE bewirkt, dass eine MQPUT- oder MQPUT1-Operation beendet wird, ohne dass die Anwendung darauf wartet, dass der Warteschlangenmanager den Aufruf beendet. Durch Angabe dieser Option kann die Nachrichtenübermittlungsleistung verbessert werden, insbesondere für Anwendungen mit Clientbindungen. Eine Anwendung kann mithilfe des MQSTAT-Verbs in regelmäßigen Abständen überprüfen, ob während eines vorherigen asynchronen Aufrufs ein Fehler aufgetreten ist.

Bei Angabe dieser Option sind nur folgende Felder im MQMD garantiert ausgefüllt:

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Wenn MQPMO_NEW_MSG_ID und/oder MQPMO_NEW_CORREL_ID als Optionen angegeben werden, sind zusätzlich auch die zurückgegebenen Felder MsgId und CorrelId ausgefüllt. (MQPMO_NEW_MSG_ID kann implizit durch die Angabe eines leeren MsgId-Feldes angegeben werden).

Es werden nur die oben genannten Felder ausgefüllt. Andere Informationen, die normalerweise in der MQMD- oder MQPMO-Struktur zurückgegeben würden, sind nicht definiert.

Bei Anforderung einer asynchronen PUT-Antwort für MQPUT1 sind ResolvedQName und ResolvedQMgrName, die in der MQOD-Struktur zurückgegeben werden, nicht definiert.

Bei Anforderung einer asynchronen PUT-Antwort für MQPUT oder MQPUT1 bedeuten ein Beendigungscode MQCC_OK und ein Ursachencode MQRC_NONE nicht notwendigerweise, dass die Nachricht erfolgreich in eine Warteschlange eingereiht wurde. Wenn Sie eine MQI-Anwendung entwickeln, die eine asynchrone PUT-Antwort verwendet und eine Bestätigung anfordert, dass Nachrichten in eine Warteschlange eingereiht wurden, müssen Sie sowohl die Beendigungs- als auch die Ursachencodes aus den PUT-Operationen überprüfen und außerdem mithilfe von MQSTAT asynchrone Fehlerinformationen abfragen.

Obwohl der Erfolg oder Fehlschlag jedes einzelnen MQPUT- oder MQPUT1-Aufrufs möglicherweise nicht unverzüglich zurückgegeben wird, kann der erste Fehler, der unter einem asynchronen Aufruf auftrat, später durch einen Aufruf an MQSTAT ermittelt werden.

Wenn eine persistente Nachricht unter Synchronisationspunktverarbeitung bei Verwendung einer asynchronen PUT-Antwort nicht zugestellt wird und Sie versuchen, die Transaktion festzuschreiben, schlägt die Festschreibung fehl und die Transaktion wird mit Beendigungscode MQCC_FAILED und Ursachencode MQRC_BACKED_OUT zurückgesetzt. Die Anwendung kann MQSTAT aufrufen, um die Ursache eines vorhergehenden MQPUT- oder MQPUT1-Fehlers zu ermitteln.

MQPMO_SYNC_RESPONSE
Die Angabe dieses PUT-Antworttyps stellt sicher, dass die MQPUT- oder MQPUT1-Operation immer synchron ausgegeben wird. Wenn die PUT-Operation erfolgreich ist, sind alle Felder im MQMD und MQPMO ausgefüllt.

Diese Option stellt eine synchrone Antwort sicher, unabhängig vom standardmäßigen PUT-Antwortwert, der im Warteschlangen- oder Themenobjekt definiert ist.

MQPMO_RESPONSE_AS_Q_DEF
Wenn dieser Wert für einen MQPUT-Aufruf angegeben ist, wird der verwendete PUT-Antworttyp dem DEFPRESP-Wert entnommen, der für die Warteschlange angegeben wurde, als sie das erste Mal von der Anwendung geöffnet wurde. Wenn eine Clientanwendung mit einem Warteschlangenmanager einer früheren Version als 7.0 verbunden ist, verhält sie sich so, als wäre MQPMO_SYNC_RESPONSE angegeben.

Wenn diese Option für einen MQPUT1-Aufruf angegeben ist, bleibt der Wert des Attributs DEFPRESP unbekannt, bis die Anforderung an den Server gesendet wird. Wenn im MQPUT1-Aufruf MQPMO_SYNCPOINT angegeben ist, verhält er sich standardmäßig wie bei MQPMO_ASYNC_RESPONSE, und wenn MQPMO_NO_SYNCPOINT angegeben ist, verhält er sich standardmäßig wie bei MQPMO_SYNC_RESPONSE. Sie können dieses Standardverhalten jedoch außer Kraft setzen, indem Sie die Eigenschaft Put1DefaultAlwaysSync in der Clientkonfigurationsdatei festlegen (siehe Zeilengruppe CHANNELS in der Clientkonfigurationsdatei).

MQPMO_RESPONSE_AS_TOPIC_DEF
MQPMO_RESPONSE_AS_TOPIC_DEF ist ein Synonym für MQPMO_RESPONSE_AS_Q_DEF zur Verwendung mit Themenobjekten.

Sonstige Optionen. Die folgenden Optionen steuern die Berechtigungsprüfung, die Vorgehensweise bei einer Versetzung des Warteschlangenmanagers in den Quiescemodus und die Auflösung von Namen von Warteschlangen und Warteschlangenmanagern:

MQPMO_ALTERNATE_USER_AUTHORITY
MQPMO_ALTERNATE_USER_AUTHORITY gibt an, dass das Feld *AlternateUserId* im Parameter *ObjDesc* des MQPUT1-Aufrufs eine Benutzer-ID enthält, die zur Prüfung der Berechtigung zum Einreihen von Nachrichten in die Warteschlange verwendet wird. Der Aufruf kann nur erfolgreich ausgeführt werden, wenn *AlternateUserId* berechtigt ist, die Warteschlange mit den angegebenen Optionen zu öffnen, unabhängig davon, ob die Benutzer-ID, unter der die Anwendung ausgeführt wird, dazu berechtigt ist. (Dies gilt jedoch nicht für angegebene Kontextoptionen, die immer anhand der Benutzer-ID überprüft werden, unter der die Anwendung ausgeführt wird.)

Diese Option ist nur mit dem Aufruf MQPUT1 gültig.

MQPMO_FAIL_IF_QUIESCING
Diese Option erzwingt ein Fehlschlagen des MQPUT- oder MQPUT1-Aufrufs, wenn sich der Warteschlangenmanager im Quiescemodus befindet.

Unter z/OSerzwingt diese Option auch das Fehlschlagen des MQPUT-oder MQPUT1 -Aufrufs, wenn sich die Verbindung (für eine CICS -oder IMS -Anwendung) im Stilllegungsstatus befindet.

Der Aufruf gibt Beendigungscode MQCC_FAILED mit Ursachencode MQRC_Q_MGR_QUIESCING oder MQRC_CONNECTION_QUIESCING zurück.

MQPMO_RESOLVE_LOCAL_Q
Verwenden Sie diese Option, um *ResolvedQName* in der MQPMO-Struktur mit dem Namen der lokalen Warteschlange, in die die Nachricht eingereiht wird, und *ResolvedQMgrName* mit dem Namen des lokalen Warteschlangenmanagers zu füllen, der die lokale Warteschlange enthält. Weitere Informationen zu MQPMO_RESOLVE_LOCAL_Q finden Sie im Abschnitt MQOO_RESOLVE_LOCAL_Q.

Wenn Sie berechtigt sind, Nachrichten in eine Warteschlange einzureihen, dann besitzen Sie auch die erforderliche Berechtigung, dieses Flag im MQPUT-Aufruf anzugeben; eine Sonderberechtigung wird nicht benötigt.

Standardoption. Wenn Sie keine der beschriebenen Optionen benötigen, verwenden Sie folgende Option:

MQPMO_NONE
Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an. MQPMO_NONE dient dazu, die Programmdokumentation zu unterstützen, und sollte nicht zusammen mit anderen Optionen verwendet werden. Da sie jedoch den Wert null hat, kann ihre Verwendung nicht erkannt werden.

MQPMO_NONE ist ein Eingabefeld. Der Anfangswert des Feldes *Options* ist MQPMO_NONE.

OriginalMsgHandle (MQHMSG)

Dies ist eine optionale Kennung für eine Nachricht. Sie wurde eventuell zuvor aus einer Warteschlange abgerufen. Die Verwendung dieser Kennung ist vom Wert des Felds *Action* abhängig; weitere Informationen siehe unter NewMsgHandle.

Der Inhalt der ursprünglichen Nachrichtenkennung wird durch den **MQPUT**- oder **MQPUT1**-Aufruf nicht geändert.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist **MQHM_NONE**. Dieses Feld wird ignoriert, wenn Version kleiner als **MQPMO_VERSION_3** ist.

PubLevel (MQLONG)
Der Anfangswert dieses Feldes ist 9. Die Höhe der Subskription, die für diese Veröffentlichung bestimmt ist. Nur die Subskriptionen mit der höchsten SubLevel, die kleiner-gleich diesem Wert ist, empfangen diese Veröffentlichung. Dieser Wert muss im Bereich zwischen 0 und 9 liegen; dabei steht 0 für die niedrigste Ebene. Wenn es sich jedoch um eine ständige Veröffentlichung handelt, ist diese für Subskribenten auf höheren Ebenen nicht mehr verfügbar, da sie auf PubLevel 1 erneut veröffentlicht wird.

Weitere Informationen finden Sie unter Veröffentlichungen abfängt .

PutMsgRecFields (MQLONG)

Dieses Feld enthält Flags, die angeben, welche MQPMR-Felder in den von der Anwendung bereitgestellten Nachrichteneinreihungssätzen vorhanden sind. Verwenden Sie *PutMsgRecFields* nur dann, wenn die Nachricht bei einer Verteilerliste eingereiht wird. Das Feld wird ignoriert, falls *RecsPresent* null ist oder falls beide, *PutMsgRecOffset* und *PutMsgRecPtr*, null sind.

Für vorhandene Felder verwendet der Warteschlangenmanager für jedes Ziel die Werte aus den Feldern im entsprechenden Datensatz der einzureihenden Nachricht. Für nicht vorhandene Felder verwendet der Warteschlangenmanager die Werte der MQMD-Struktur.

Verwenden Sie mindestens eine der folgenden Flags, um anzuzeigen, welche Felder in den Nachrichteneinreihungssätzen vorhanden sind:

MQPMRF_MSG_ID
Nachrichten-ID-Feld ist vorhanden.

MQPMRF_CORREL_ID
Korrelations-ID-Feld ist vorhanden.

MQPMRF_GROUP_ID
Gruppen-ID-Feld ist vorhanden.

MQPMRF_FEEDBACK
Feedbackfeld ist vorhanden.

MQPMRF_ACCOUNTING_TOKEN
Feld für das Abrechnungstoken vorhanden.

Geben Sie, wenn Sie dieses Flag angeben, entweder MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT im *Options*-Feld an; wird diese Bedingung nicht erfüllt, schlägt der Aufruf mit dem Ursachencode MQRC_PMO_RECORD_FLAGS_ERROR fehl.

Wenn keine MQPMR-Felder vorhanden sind, kann Folgendes angegeben werden:

MQPMRF_NONE
Es sind keine Felder mit Nachrichteneinreihungssätzen vorhanden.

Wenn dieser Wert angegeben wird, muss entweder *RecsPresent* null sein oder beide, *PutMsgRecOffset* und *PutMsgRecPtr*, müssen null sein.

MQPMRF_NONE dient zur Unterstützung der Programmdokumentation. Diese Konstante ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

Wenn *PutMsgRecFields* Flags enthält, die nicht gültig sind, oder Nachrichteneinreihungssätze bereitgestellt werden, *PutMsgRecFields* jedoch den Wert MQPMRF_NONE hat, schlägt der Aufruf mit dem Ursachencode MQRC_PMO_RECORD_FLAGS_ERROR fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist MQPMRF_NONE. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG)

Dies ist die in Byte angegebene relative Adresse der ersten MQPMR-Datensätze der einzureihenden Nachricht ab dem Anfang der MQPMO-Struktur. Der Offset kann positiv oder negativ sein. *PutMsgRecOffset* wird nur verwendet, wenn die Nachricht in eine Verteilerliste eingereiht wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Wenn die Nachricht in eine Verteilerliste eingereiht wird, kann eine Feldgruppe mit mindestens einem MQPMR-Nachrichteneinreihungssatz bereitgestellt werden, um für jede Zieladresse bestimmte Eigenschaften der Nachricht individuell anzugeben; bei diesen Eigenschaften handelt es sich um:

- Nachrichten-ID
- Korrelations-ID
- Gruppen-ID
- Rückmeldungswert
- Abrechnung

Sie müssen nicht alle dieser Eigenschaften angeben, achten Sie aber darauf, dass Sie, unabhängig davon, welche Teilmenge sie verwenden, die Felder in der richtigen Reihenfolge angeben. Weitere Informationen finden Sie in der Beschreibung der MQPMR-Struktur.

Normalerweise muss die Zahl der Nachrichteneinreihungssätze der der Objektdatensätze entsprechen, die der MQOD angibt, wenn die Verteilerliste geöffnet wird; jeder Nachrichteneinreihungssatz stellt die Nachrichteneigenschaften für die vom Objektdatensatz ermittelte Warteschlange bereit. Warteschlangen in der Verteilerliste, die sich nicht öffnen, müssen dennoch an den geeigneten Positionen in der Feldgruppe Nachrichteneinreihungssätze zugeordnet werden, auch wenn in diesem Fall die Nachrichteneigenschaften ignoriert werden.

Die Zahl der Nachrichteneinreihungssätze kann sich von der Zahl der Objektdatensätze unterscheiden. Wenn die Zahl der Nachrichteneinreihungssätze die Zahl der Objektdatensätze unterschreitet, werden die Nachrichteneigenschaften der Zieladressen, die über keine Nachrichteneinreihungssätze verfügen, von den entsprechenden Feldern im Nachrichtendeskriptor MQMD übernommen. Falls die Zahl der Nachrichteneinreihungssätze die der Objektdatensätze überschreitet, wird der Überschuss nicht verwendet, allerdings muss es immer noch möglich bleiben, darauf zuzugreifen. Nachrichteneinreihungssätze sind optional, aber wenn sie zur Verfügung gestellt werden, muss es *RecsPresent* für sie geben.

Stellen Sie die Nachrichteneinreihungssätze auf ähnliche Art wie die Objektdatensätze in MQOD bereit, indem Sie entweder einen Offset in *PutMsgRecOffset* angeben oder eine Adresse in *PutMsgRecPtr*; weitere Informationen dazu, wie Sie dazu vorgehen müssen, finden Sie in der Beschreibung des *ObjectRecOffset*-Feldes im Abschnitt über „MQOD - Objektdeskriptor" auf Seite 463.

Es kann jedoch nur eines der Felder, *PutMsgRecOffset* oder *PutMsgRecPtr*, verwendet werden; der Aufruf schlägt mit dem Ursachencode MQRC_PUT_MSG_RECORDS_ERROR fehl, wenn beide ungleich null sind.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR)

Dies ist die Adresse des ersten MQPMR-Nachrichteneinreihungssatzes. Verwenden Sie *PutMsgRecPtr* nur dann, wenn die Nachricht bei einer Verteilerliste eingereiht wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Verwenden Sie entweder *PutMsgRecPtr* oder *PutMsgRecOffset*, um die Nachrichteneinreihungssätze anzugeben, aber verwenden Sie nicht beide; weitere Einzelheiten finden Sie in der Beschreibung des *PutMsgRecOffset*-Feldes oben. Wenn Sie *PutMsgRecPtr* nicht verwenden, setzen Sie das Feld auf den Nullzeiger oder auf null Bytes.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

Anmerkung: Bei Plattformen, bei denen die Programmiersprache den Datentyp Zeiger nicht unterstützt, wird das Feld als Bytefolge der entsprechenden Länge deklariert, wobei der Anfangswert die vollständig auf null gesetzte Bytefolge ist.

RecsPresent (MQLONG)

Dies ist die Anzahl an MQPMR-Datensätzen der einzureihenden Nachricht bzw. an MQRR-Antwortdatensätzen, die durch die Anwendung bereitgestellt wurden. Diese Anzahl kann nur größer als null sein, wenn die Nachricht in eine Verteilerliste eingereiht wird. Nachrichteneinreihungssätze und Antwortdatensätze sind optional; die Anwendung muss keine Datensätze bereitstellen oder sie kann nur Datensätze eines einzigen Typs bereitstellen. Wenn die Anwendung jedoch beide Datensatztypen bereitstellt, muss sie auch *RecsPresent*-Datensätze beider Typen zur Verfügung stellen.

Der Wert von *RecsPresent* muss nicht der Anzahl der Zieladressen der Verteilerliste entsprechen. Werden zu viele Datensätze bereitgestellt, wird der Überschuss nicht verwendet; wenn zu wenig Datensätze bereitgestellt werden, werden Standardwerte für die Nachrichteneigenschaften derjeniger Zieladressen

verwendet, die nicht über Nachrichteneinreihungssätze verfügen (weitere Informationen finden Sie im Abschnitt über *PutMsgRecOffset*).

Wenn *RecsPresent* kleiner als null ist oder wenn das Feld größer als null ist, die Nachricht aber nicht in einer Verteilerliste eingereiht wird, schlägt der Aufruf mit dem Ursachencode MQRC_RECS_PRESENT_ERROR fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

ResolvedQMgrName (MQCHAR48)

Dies ist der Name des Ziel-Warteschlangenmanagers nach der Namensauflösung durch den lokalen Warteschlangenmanager. Der zurückgegebene Name ist der Name des Warteschlangenmanagers, der der Eigner der Warteschlange ist, die von *ResolvedQName* ermittelt wurde, und kann der Name des lokalen Warteschlangenmanagers sein.

Wenn *ResolvedQName* eine gemeinsam genutzte Warteschlange ist, deren Eigner die Gruppe mit gemeinsamer Warteschlange ist, zu der der lokale Warteschlangenmanager gehört, dann ist *ResolvedQMgrName* der Name der Gruppe mit gemeinsamer Warteschlange. Wenn die Warteschlange einer anderen Gruppe mit gemeinsamer Warteschlange zugeordnet ist, kann *ResolvedQName* der Name der Gruppe mit gemeinsamer Warteschlange oder der Name eines Warteschlangenmanagers sein, der ein Mitglied der Gruppe mit gemeinsamer Warteschlange ist (die Art des zurückgegebenen Werts ist von den Warteschlangendefinitionen des lokalen Warteschlangenmanagers abhängig).

Ein belegter Wert wird nur dann zurückgegeben, wenn das Objekt eine einzelne Warteschlange ist; handelt es sich bei dem Objekt um eine Verteilerliste oder ein Thema, ist der zurückgegebene Wert nicht definiert.

Dies ist ein Ausgabefeld. Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ResolvedQName (MQCHAR48)

Dies ist der Name der Zielwarteschlange nach der Namensauflösung durch den lokalen Warteschlangenmanager. Der zurückgegebene Name gibt den Namen einer Warteschlange an, so wie sie auf dem mit *ResolvedQMgrName* angegebenen Warteschlangenmanager definiert ist.

Ein belegter Wert wird nur dann zurückgegeben, wenn das Objekt eine einzelne Warteschlange ist; handelt es sich bei dem Objekt um eine Verteilerliste oder ein Thema, ist der zurückgegebene Wert nicht definiert.

Dies ist ein Ausgabefeld. Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

ResponseRecOffset (MQLONG)

Dies ist die in Byte angegebene relative Adresse des ersten MQPMR-Antwortdatensatzes ab dem Anfang der MQPMO-Struktur. Der Offset kann positiv oder negativ sein. *ResponseRecOffset* wird nur verwendet, wenn die Nachricht in eine Verteilerliste eingereiht wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Wenn Sie die Nachricht in eine Verteilerliste einreihen, können Sie eine Feldgruppe mit mindestens einem MQRR-Antwortdatensatz bereitstellen, um zum einen die Warteschlangen zu ermitteln, an die die Nachricht nicht erfolgreich gesendet werden konnte (*CompCode*-Feld in MQRR) und zum anderen den Grund für das Fehlschlagen (*Reason*-Feld in MQRR). Der Grund dafür, dass die Nachricht nicht gesendet wurde, kann entweder sein, dass die Warteschlange nicht geöffnet werden konnte oder dass die PUT-Operation fehlschlug. Der Warteschlangenmanager setzt nur dann die Antwortdatensätze, wenn das Ergebnis des Aufrufs heterogen ist, d. h. wenn einige Nachrichten erfolgreich gesendet wurden, während andere fehlschlugen, bzw. wenn alle fehlschlugen, allerdings aus unterschiedlichen Gründen; der Ursachencode MQRC_MULTIPLE_REASONS des Aufrufs weist darauf hin. Wenn derselbe Ursachencode für alle Warteschlangen zutrifft, wird diese Ursache im *Reason*-Parameter des MQPUT- oder des MQPUT1-Aufrufs wiedergegeben und die Antwortdatensätze werden nicht gesetzt.

Normalerweise entspricht die Zahl der Antwortdatensätze der der Objektdatensätze, die der MQOD angibt, wenn die Verteilerliste geöffnet wird; nötigenfalls wird jeder Antwortdatensatz auf den Beendigungs- und auf den Ursachencode für die Einreihung in die Warteschlange gesetzt, die von dem entsprechenden Objektdatensatz ermittelt wurden. Warteschlangen in der Verteilerliste, die sich nicht öffnen, müssen auch dann Antwortdatensätze an den geeigneten Positionen in der Feldgruppe zugeordnet werden, wenn sie auf Beendigungs- und Ursachencode gesetzt werden, wie sie sich aus der Operation zum Öffnen ergeben statt aus der PUT-Operation.

Die Zahl der Antwortdatensätze kann sich von der Zahl der Objektdatensätze unterscheiden. Falls die Zahl der Antwortdatensätze geringer ist als die Zahl der Objektdatensätze, kann die Anwendung eventuell nicht alle Zieladressen ermitteln, bei denen die PUT-Operation fehlschlug, oder sie kann die Gründe für das Fehlschlagen nicht ermitteln. Wenn mehr Antwortdatensätze vorhanden sind als Objektdatensätze, so werden die überschüssigen Datensätze nicht verwendet (obwohl es weiterhin möglich sein muss, auf sie zuzugreifen). Antwortdatensätze sind optional, aber wenn sie zur Verfügung gestellt werden, muss es *RecsPresent* für sie geben.

Stellen Sie die Antwortdatensätze auf ähnliche Weise wie die Objektdatensätze in MQOD bereit, indem Sie entweder einen Offset in *ResponseRecOffset* angeben oder eine Adresse in *ResponseRecPtr*; weitere Informationen dazu, wie Sie dazu vorgehen müssen, finden Sie in der Beschreibung des *ObjectRecOffset*-Feldes im Abschnitt über „MQOD - Objektdeskriptor" auf Seite 463. Setzen Sie jedoch nur eines der Felder, *ResponseRecOffset* oder *ResponseRecPtr*, auf ungleich null; der Aufruf schlägt mit dem Ursachencode MQRC_RESPONSE_RECORDS_ERROR fehl, wenn beide ungleich null sind.

Beim MQPUT1-Aufruf muss dieses Feld null sein. Dies liegt daran, dass die Antwortinformation, falls sie angefordert wird, in den Antwortdatensätzen zurückgegeben wird, die vom Objektdeskriptor MQOD angegeben wird.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

ResponseRecPtr (MQPTR)

Dies ist die Adresse des ersten MQRR-Antwortdatensatzes. *ResponseRecPtr* wird nur verwendet, wenn die Nachricht in eine Verteilerliste eingereiht wird. Das Feld wird ignoriert, falls *RecsPresent* null ist.

Verwenden Sie entweder *ResponseRecPtr* oder *ResponseRecOffset*, um die Antwortdatensätze anzugeben, aber verwenden Sie nicht beide; weitere Einzelheiten finden Sie in der Beschreibung des *ResponseRecOffset*-Feldes oben. Wenn Sie *ResponseRecPtr* nicht verwenden, setzen Sie das Feld auf den Nullzeiger oder auf null Bytes.

Für den MQPUT1-Aufruf muss dieses Feld auf den Nullzeiger oder auf null Byte gesetzt sein. Dies liegt daran, dass die Antwortinformation, falls sie angefordert wird, in den Antwortdatensätzen zurückgegeben wird, die vom Objektdeskriptor MQOD angegeben wird.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge. Dieses Feld wird ignoriert, wenn die *Version* älter ist als MQPMO_VERSION_2.

Anmerkung: Bei Plattformen, bei denen die Programmiersprache den Datentyp Zeiger nicht unterstützt, wird das Feld als Bytefolge der entsprechenden Länge deklariert, wobei der Anfangswert die vollständig auf null gesetzte Bytefolge ist.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQPMO_STRUC_ID
ID für die Struktur der Nachrichteneinreihungsoptionen

Für die Programmiersprache C ist auch die Konstante MQPMO_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie die Konstante MQPMO_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQPMO_STRUC_ID.

Timeout (MQLONG)

Dies ist ein reserviertes Feld, dessen Wert nicht von Bedeutung ist. Der Anfangswert dieses Felds ist -1.

UnknownDestCount (MQLONG)

Dies ist die Anzahl der Nachrichten, welche mit dem aktuellen MQPUT- oder MQPUT1-Aufruf erfolgreich an Warteschlangen in der Verteilerliste gesendet werden konnten, die als ferne Warteschlangen aufgelöst sind. Nachrichten, die der Warteschlangenmanager vorübergehend in Verteilerlistenform beibehält, werden als die Anzahl der in diesen Verteilerlisten enthaltenen einzelnen Ziele gezählt. Dieses Feld wird auch gesetzt, wenn eine Nachricht in einer einzelnen Warteschlange eingereiht wird, die keiner Verteilerliste angehört.

Dies ist ein Ausgabefeld. Der Anfangswert dieses Feldes ist 0. Dieses Feld ist nicht festgelegt, wenn die *Version* niedriger ist als MQPMO_VERSION_1.

Dieses Feld ist bei z/OS nicht definiert, da keine Verteilerlisten unterstützt werden.

Version (MQLONG)
Strukturversionsnummer.

Folgende Werte sind möglich:

MQPMO_VERSION_1
Version-1 Nachrichteneinreihungsoptionsstruktur.

Diese Option wird in allen Umgebungen unterstützt.

MQPMO_VERSION_2
Version-2 Nachrichteneinreihungsoptionsstruktur.

Diese Version wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

MQPMO_VERSION_3
Version-3 Nachrichteneinreihungsoptionsstruktur.

Diese Option wird in allen Umgebungen unterstützt.

Nur in der neueren Version der Struktur verwendete Felder werden in den Feldbeschreibungen entsprechend gekennzeichnet. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQPMO_CURRENT_VERSION
Aktuelle Version der Optionsstruktur für die Nachrichteneinreihung.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQPMO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQPMO

Tabelle 529. Anfangswerte von Feldern in MQPMO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQPMO_STRUC_ID	'PMO¬'
Version	MQPMO_VERSION_1	1
Options	MQPMO_NONE	0
Timeout	--	-1
Context	--	0
KnownDestCount	--	0
UnknownDestCount	--	0
InvalidDestCount	--	0

Tabelle 529. Anfangswerte von Feldern in MQPMO (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
ResolvedQName	--	Nullzeichenfolge oder Leerzeichen.
ResolvedQMgrName	--	Nullzeichenfolge oder Leerzeichen.
RecsPresent	--	0
PutMsgRecFields	MQPMRF_NONE	0
PutMsgRecOffset	--	0
ResponseRecOffset	--	0
PutMsgRecPtr	--	Nullzeiger oder Null Byte
ResponseRecPtr	--	Nullzeiger oder Null Byte
OriginalMsgHandle	MQHM_NONE	0
NewMsgHandle	MQHM_NONE	0
Action	MQACTP_NEW	0
PubLevel	--	9

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQPMO_DEFAULT die oben angegebenen Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQLONG    Options;            /* Options that control the action of
                                   MQPUT and MQPUT1 */
  MQLONG    Timeout;            /* Reserved */
  MQHOBJ    Context;            /* Object handle of input queue */
  MQLONG    KnownDestCount;     /* Number of messages sent
                                   successfully to local queues */
  MQLONG    UnknownDestCount;   /* Number of messages sent
                                   successfully to remote queues */
  MQLONG    InvalidDestCount;   /* Number of messages that could not
                                   be sent */
  MQCHAR48  ResolvedQName;      /* Resolved name of destination
                                   queue */
  MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination queue
                                   manager */
  /* Ver:1 */
  MQLONG    RecsPresent;        /* Number of put message records or
                                   response records present */
  MQLONG    PutMsgRecFields;    /* Flags indicating which MQPMR fields
                                   are present */
  MQLONG    PutMsgRecOffset;    /* Offset of first put message record
                                   from start of MQPMO */
  MQLONG    ResponseRecOffset;  /* Offset of first response record
```

```
                                   from start of MQPMO */
  MQPTR     PutMsgRecPtr;       /* Address of first put message
                                   record */
  MQPTR     ResponseRecPtr;     /* Address of first response record */
  /* Ver:2 */
  MQHMSG    OriginalMsgHandle;  /* Original message handle */
  MQHMSG    NewMsgHandle;       /* New message handle */
  MQLONG    Action;             /* The action being performed */
  MQLONG    PubLevel;           /* Subscription level */
  /* Ver:3 */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQPMO based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 Options           fixed bin(31), /* Options that control the action
                                        of MQPUT and MQPUT1 */
  3 Timeout           fixed bin(31), /* Reserved */
  3 Context           fixed bin(31), /* Object handle of input queue */
  3 KnownDestCount    fixed bin(31), /* Number of messages sent
                                        successfully to local queues */
  3 UnknownDestCount  fixed bin(31), /* Number of messages sent
                                        successfully to remote queues */
  3 InvalidDestCount  fixed bin(31), /* Number of messages that could
                                        not be sent */
  3 ResolvedQName     char(48),      /* Resolved name of destination
                                        queue */
  3 ResolvedQMgrName  char(48),      /* Resolved name of destination
```

```
                                       queue manager */
  3 RecsPresent       fixed bin(31), /* Number of put message records or
                                       response records present */
  3 PutMsgRecFields   fixed bin(31), /* Flags indicating which MQPMR
                                       fields are present */
  3 PutMsgRecOffset   fixed bin(31), /* Offset of first put message
                                       record from start of MQPMO */
  3 ResponseRecOffset fixed bin(31), /* Offset of first response record
                                       from start of MQPMO */
  3 PutMsgRecPtr      pointer,       /* Address of first put message
                                       record */
  3 ResponseRecPtr    pointer,       /* Address of first response
                                       record */
  3 OriginalMsgHandle fixed bin(63), /* Original message handle */
  3 NewMsgHandle      fixed bin(63); /* New message handle */
  3 Action            fixed bin(31); /* The action being performed */
  3 PubLevel          fixed bin(31); /* Publish level */
```

Deklaration in High Level Assembler

```
MQPMO                   DSECT
MQPMO_STRUCID           DS   CL4   Structure identifier
MQPMO_VERSION           DS   F     Structure version number
MQPMO_OPTIONS           DS   F     Options that control the action of
*                                  MQPUT and MQPUT1
MQPMO_TIMEOUT           DS   F     Reserved
MQPMO_CONTEXT           DS   F     Object handle of input queue
MQPMO_KNOWNDESTCOUNT    DS   F     Number of messages sent successfully
*                                  to local queues
MQPMO_UNKNOWNDESTCOUNT  DS   F     Number of messages sent successfully
*                                  to remote queues
MQPMO_INVALIDDESTCOUNT  DS   F     Number of messages that could not be
*                                  sent
MQPMO_RESOLVEDQNAME     DS   CL48  Resolved name of destination queue
MQPMO_RESOLVEDQMGRNAME  DS   CL48  Resolved name of destination queue
*                                  manager
MQPMO_RECSPRESENT       DS   F     Number of put message records or
*                                  response records present
MQPMO_PUTMSGRECFIELDS   DS   F     Flags indicating which MQPMR
*                                  fields are present
MQPMO_PUTMSGRECOFFSET   DS   F     Offset of first put message record
*                                  from start of MQPMO
MQPMO_RESPONSERECOFFSET DS   F     Offset of first response record
*                                  from start of MQPMO
MQPMO_PUTMSGRECPTR      DS   F     Address of first put message
*                                  record
MQPMO_RESPONSERECPTR    DS   F     Address of first response record
MQPMO_ORIGINALMSGHANDLE DS   D     Original message handle
MQPMO_NEWMSGHANDLE      DS   D     New message handle
MQPMO_ACTION            DS   F     The action being performed
MQPMO_PUBLEVEL          DS   F     Publish level
*
MQPMO_LENGTH            EQU  *-MQPMO
                        ORG  MQPMO
MQPMO_AREA              DS   CL(MQPMO_LENGTH)
```

Deklaration in Visual Basic

```
Type MQPMO
  StrucId           As String*4  'Structure identifier'
  Version           As Long      'Structure version number'
  Options           As Long      'Options that control the action of'
                                 'MQPUT and MQPUT1'
  Timeout           As Long      'Reserved'
  Context           As Long      'Object handle of input queue'
  KnownDestCount    As Long      'Number of messages sent successfully'
                                 'to local queues'
  UnknownDestCount  As Long      'Number of messages sent successfully'
                                 'to remote queues'
  InvalidDestCount  As Long      'Number of messages that could not be'
                                 'sent'
  ResolvedQName     As String*48 'Resolved name of destination queue'
  ResolvedQMgrName  As String*48 'Resolved name of destination queue'
                                 'manager'
  RecsPresent       As Long      'Number of put message records or'
                                 'response records present'
```

```
   PutMsgRecFields   As Long      'Flags indicating which MQPMR fields'
                                 'are present'
   PutMsgRecOffset   As Long      'Offset of first put message record'
                                 'from start of MQPMO'
   ResponseRecOffset As Long      'Offset of first response record from'
                                 'start of MQPMO'
   PutMsgRecPtr      As MQPTR     'Address of first put message record'
   ResponseRecPtr    As MQPTR     'Address of first response record'
End Type
```

MQPMR – Datensatz für das Einreihen von Nachrichten

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 530. Felder für MQPMR

Feld	Beschreibung	Thema
MsgId	Nachrichten-ID	MsgId
CorrelId	Korrelations-ID	CorrelId
GroupId	Gruppen-ID	GroupId
Feedback	Rückmeldung oder Ursachencode	Rückmeldung
AccountingToken	Abrechnung	AccountingToken

Überblick zu MQPMR

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Zweck: Verwenden Sie die MQPMR-Struktur, um verschiedene Nachrichteneigenschaften für ein einzelnes Ziel anzugeben, wenn eine Nachricht in eine Verteilerliste eingereiht wird. MQPMR ist eine Ein-/Ausgabe-Struktur für MQPUT- und MQPUT1-Aufrufe.

Zeichensatz und Codierung: Der Zeichensatz der Daten in MQPMR muss dem entsprechen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, die Codierung der des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angebeben ist. Wenn die Anwendung allerdings als MQ-Client ausgeführt wird, müssen Zeichensatz und Codierung der Struktur des Clients entsprechen.

Verwendung: Wenn Sie ein Array dieser Strukturen beim MQPUT- oder MQPUT1-Aufruf bereitstellen, können Sie unterschiedliche Werte für jede Zielwarteschlange in einer Verteilerliste angeben. Einige Felder sind nur Eingabefelder, andere Ein-/Ausgabefelder.

Anmerkung: Diese Struktur ist ungewöhnlich, da sie keinen festen Aufbau hat. Die Felder in dieser Struktur sind optional, das Vorhandensein oder Fehlen des jeweiligen Feldes wird durch die Flags im Feld *PutMsgRecFields* in MQPMO angegeben. Vorhandene Felder ***müssen in folgender Reihenfolge angeordnet sein***:

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Nicht vorhandene Felder belegen keinen Speicherplatz im Datensatz.

Da MQPMR kein festgelegtes Layout hat, wird im Header und in den COPY- bzw. INCLUDE-Dateien für die unterstützten Programmiersprachen keine Definition bereitgestellt. Der Anwendungsprogrammierer muss eine Deklaration erstellen, die die Felder umfasst, die die Anwendung erfordert, und die Flags in *PutMsgRecFields* festlegen, um die vorhandenen Felder anzugeben.

Felder für MQPMR

Die MQPMR-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

AccountingToken (MQBYTE32)

Dies ist das Abrechnungstoken, das für die Nachricht verwendet wird, die an die Warteschlange mit dem Namen gesendet wird, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde. Es wird auf die gleiche Weise verarbeitet wie das Feld *AccountingToken* in MQMD für das Einreihen in eine einzelne Warteschlange. Weitere Hinweise zum Inhalt dieses Felds finden Sie in der Beschreibung zu *AccountingToken* in „MQMD - Nachrichtendeskriptor" auf Seite 399.

Ist dieses Feld nicht vorhanden, so wird der Wert im MQMD verwendet.

Dies ist ein Eingabefeld.

CorrelId (MQBYTE24)

Dies ist die Korrelations-ID, die für die Nachricht verwendet wird, die an die Warteschlange mit dem Namen gesendet wird, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde. Sie wird auf die gleiche Weise verarbeitet wie das Feld *CorrelId* in MQMD für das Einreihen in eine einzelne Warteschlange.

Wenn dieses Feld nicht im MQPMR-Datensatz vorhanden ist oder es weniger MQPMR-Datensätze als Ziele gibt, wird der Wert in MQMD für die Ziele verwendet, die keinen MQPMR-Datensatz mit dem Feld *CorrelId* enthalten.

Wenn MQPMO_NEW_CORREL_ID angegeben wird, wird eine *einzelne* neue Korrelations-ID erstellt und für alle Ziele in der Verteilerliste verwendet, unabhängig davon, ob sie MQPMR-Datensätze enthalten. Dies unterscheidet sich von der Verarbeitung von MQPMO_NEW_MSG_ID (siehe Feld *MsgId*).

Dies ist ein Ein-/Ausgabefeld.

Feedback (MQLONG)

Dies ist der Rückmeldungscode, der für die Nachricht verwendet wird, die an die Warteschlange mit dem Namen gesendet wird, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde. Er wird auf die gleiche Weise verarbeitet wie das Feld *Feedback* in MQMD für das Einreihen in eine einzelne Warteschlange.

Ist dieses Feld nicht vorhanden, so wird der Wert im MQMD verwendet.

Dies ist ein Eingabefeld.

GroupId (MQBYTE24)

Dies ist die Gruppen-ID, die für die Nachricht verwendet wird, die an die Warteschlange mit dem Namen gesendet wird, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde. Sie wird auf die gleiche Weise verarbeitet wie das Feld *GroupId* in MQMD für das Einreihen in eine einzelne Warteschlange.

Wenn dieses Feld nicht im MQPMR-Datensatz vorhanden ist oder es weniger MQPMR-Datensätze als Ziele gibt, wird der Wert in MQMD für die Ziele verwendet, die keinen MQPMR-Datensatz mit dem Feld *GroupId* enthalten. Der Wert wird wie unter Physische Reihenfolge in einer Warteschlange beschrieben verarbeitet, mit den folgenden Unterschieden:

- GroupId wird anhand von QMName und einer Zeitmarke erstellt. Um eine eindeutige GroupId zu erhalten, müssen Warteschlangenmanagernamen ebenfalls eindeutig sein. Auch dürfen Sie die Systemzeit der Warteschlangenmanager-Systeme nicht zurückstellen.
- In den Fällen, in denen eine neue Gruppen-ID verwendet werden würde, generiert der Warteschlangenmanager für jedes Ziel eine andere Gruppen-ID (das heißt, zwei verschiedene Ziele können nie die gleiche Gruppen-ID haben).
- Wenn der Wert im Feld verwendet würde, schlägt der Aufruf mit dem Ursachencode MQRC_GROUP_ID_ERROR fehl.

Dies ist ein Ein-/Ausgabefeld.

MsgId (MQBYTE24)

Dies ist die Nachrichten-ID, die für die Nachricht verwendet wird, die an die Warteschlange mit dem Namen gesendet wird, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde. Sie wird auf die gleiche Weise verarbeitet wie das Feld *MsgId* in MQMD für das Einreihen in eine einzelne Warteschlange.

Wenn dieses Feld nicht im MQPMR-Datensatz vorhanden ist oder es weniger MQPMR-Datensätze als Ziele gibt, wird der Wert in MQMD für die Ziele verwendet, die keinen MQPMR-Datensatz mit dem Feld *MsgId* enthalten. Wenn dieser Wert MQMI_NONE ist, wird eine neue Nachrichten-ID für *jedes* dieser Ziele generiert (d. h., dass nicht zwei Ziele dieselbe Nachrichten-ID haben).

Wenn MQPMO_NEW_MSG_ID angegeben ist, werden neue Nachrichten-IDs für alle Ziele in der Verteilerliste erstellt, unabhängig davon, ob sie MQPMR-Datensätze enthalten. Dies unterscheidet sich von der Verarbeitung von MQPMO_NEW_CORREL_ID (siehe Feld *CorrelId*).

Dies ist ein Ein-/Ausgabefeld.

Anfangswerte und Sprachendeklarationen für MQPMR

Es gibt für diese Struktur keine definierten Anfangswerte, weil keine Strukturdeklarationen im Header und in den COPY- bzw. INCLUDE-Dateien für die unterstützten Programmiersprachen bereitgestellt werden. Die Musterdeklarationen zeigen, wie die Struktur zu deklarieren ist, wenn alle Felder erforderlich sind.

Deklaration in Programmiersprache C

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
  MQBYTE24  MsgId;            /* Message identifier */
  MQBYTE24  CorrelId;         /* Correlation identifier */
  MQBYTE24  GroupId;          /* Group identifier */
  MQLONG    Feedback;         /* Feedback or reason code */
  MQBYTE32  AccountingToken;  /* Accounting token */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQPMR based,
   3 MsgId           char(24),      /* Message identifier */
   3 CorrelId        char(24),      /* Correlation identifier */
   3 GroupId         char(24),      /* Group identifier */
   3 Feedback        fixed bin(31), /* Feedback or reason code */
   3 AccountingToken char(32);      /* Accounting token */
```

Deklaration in Visual Basic

```
Type MQPMR
```

```
   MsgId           As MQBYTE24 'Message identifier'
   CorrelId        As MQBYTE24 'Correlation identifier'
   GroupId         As MQBYTE24 'Group identifier'
   Feedback        As Long     'Feedback or reason code'
   AccountingToken As MQBYTE32 'Accounting token'
End Type
```

MQRFH - Header für Regeln und Formatierung

Dieser Abschnitt beschreibt den Regel- und Formatierungsheader, welche Felder er enthält und die Anfangswerte dieser Felder.

Überblick zu MQRFH

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQRFH-Struktur definiert das Layout des Regel- und Formatierungsheaders. Verwenden Sie diesen Header, um Zeichenfolgedaten als Name/Wert-Paare zu senden.

Formatname: MQFMT_RF_HEADER.

Zeichensatz und Codierung: Die Felder in der MQRFH-Struktur (einschließlich *NameValueString*) haben den Zeichensatz und die Codierung, die durch die Felder *CodedCharSetId* und *Encoding* in der Headerstruktur vor dem MQRFH-Header oder durch die Felder in der MQMD-Struktur angegeben sind, wenn der MQRFH am Anfang der Anwendungsnachrichtendaten steht.

Um in Warteschlangennamen gültige Zeichen bereitstellen zu können, muss der Zeichensatz Einzelbytezeichen unterstützen.

Felder für MQRFH

Die MQRFH-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)

Gibt die Zeichensatzkennung der Daten an, die auf *NameValueString* folgen; dieses Attribut bezieht sich nicht auf Zeichendaten in der MQRFH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Folgende Sonderwerte sind zulässig:

MQCCSI_INHERIT
Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

MQCCSI_INHERIT kann nicht verwendet werden, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

Encoding (MQLONG)

Gibt die numerische Codierung der Daten an, die auf *NameValueString* folgen; dieses Attribut bezieht sich nicht auf numerische Daten in der MQRFH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

Flags (MQLONG)

Folgende Werte sind zulässig:

MQRFH_NONE
Keine Flags.

Der Anfangswert dieses Felds ist MQRFH_NONE.

Format (MQCHAR8)

Gibt den Formatnamen der Daten an, die auf *NameValueString* folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds entsprechen denen für das Feld *Format* in MQMD.

Der Anfangswert dieses Felds ist MQFMT_NONE.

NameValueString (MQCHARn)

Dies ist eine Zeichenfolge variabler Länge, die Name/Wert-Paare der folgenden Form enthält:

```
name1 value1 name2 value2 name3 value3 ...
```

Namen müssen jeweils durch ein oder mehrere Leerzeichen vom angrenzenden Namen oder Wert getrennt sein. Diese Leerzeichen sind nicht signifikant. Signifikante Leerzeichen in einem Namen oder Wert müssen in Anführungszeichen eingeschlossen werden. Alle Zeichen zwischen dem öffnenden und dem entsprechenden schließenden Anführungszeichen werden als signifikant behandelt. Im folgenden Beispiel lautet der Name FAMOUS_WORDSund der Wert lautet `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Ein Name oder ein Wert kann jedes Zeichen außer dem Nullzeichen enthalten (das als Trennzeichen für *NameValueString* verwendet wird). Um die Interoperabilität zu unterstützen, kann eine Anwendung Namen auf folgende Zeichen beschränken:

- Erstes Zeichen: Alphabetisch in Groß- oder Kleinschreibung (A bis Z oder a bis z) oder Unterstrich.
- Nachfolgende Zeichen: Groß- oder Kleinbuchstaben des Alphabets, Dezimalziffern (0 bis 9), Unterstreichung, Trennstrich oder Punkt.

Wenn ein Name oder ein Wert ein oder mehrere Anführungszeichen enthält, muss der Name oder der Wert in Anführungszeichen eingeschlossen werden und jedes Anführungszeichen innerhalb der Zeichenfolge muss verdoppelt werden.

```
Famous_Words "The program displayed ""Hello World"""
```

Bei Namen und Werten muss die Groß- und Kleinschreibung berücksichtigt werden; das heißt, Klein- und Großbuchstaben sind nicht austauschbar. FAMOUS_WORDS und `Famous_Words` sind beispielsweise zwei unterschiedliche Namen.

Die Länge in Byte von *NameValueString* entspricht *StrucLength* minus MQRFH_STRUC_LENGTH_FIXED. Um Probleme bei der Konvertierung von Benutzerdaten in bestimmten Umgebungen zu vermeiden, legen Sie diese Länge als ein Vielfaches von vier fest. Füllen Sie *NameValueString* mit Leerzeichen auf diese Länge auf oder beenden Sie die Zeichenfolge früher, indem Sie nach dem letzten signifikanten Zeichen ein Nullzeichen platzieren. Das Nullzeichen und die ihm folgenden Bytes bis zur angegebenen Länge von *NameValueString* werden ignoriert.

Anmerkung: Da die Länge dieses Felds nicht festgelegt ist, fehlt es in den Deklarationen der Struktur, die für die unterstützten Programmiersprachen bereitgestellt werden.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQRFH_STRUC_ID
ID für Regel- und Formatierungsheaderstuktur.

Für die Programmiersprache C ist auch die Konstante MQRFH_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQRFH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQRFH_STRUC_ID.

StrucLength (MQLONG)

Dies ist die Länge der MQRFH-Struktur in Byte, einschließlich des Feldes *NameValueString* am Ende der Struktur. Benutzerdaten, die auf das Feld *NameValueString* folgen, sind *nicht* eingeschlossen.

Um Probleme bei der Konvertierung von Benutzerdaten in bestimmten Umgebungen zu vermeiden, muss *StrucLength* ein Vielfaches von vier sein.

Die folgende Konstante gibt die Länge der *Festkomponente* der Struktur an, d. h. die Länge ausschließlich des Feldes *NameValueString*:

MQRFH_STRUC_LENGTH_FIXED
Länge der Festkomponente der MQRFH-Struktur.

Der Anfangswert dieses Felds ist MQRFH_STRUC_LENGTH_FIXED.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQRFH_VERSION_1
Regel- und Formatierungsheaderstuktur der Version 1.

Der Anfangswert dieses Felds ist MQRFH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQRFH

Tabelle 531. Anfangswerte für Felder im MQRFH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQRFH_STRUC_ID	'RFH¬'
Version	MQRFH_VERSION_1	1
StrucLength	MQRFH_STRUC_LENGTH_FIXED	32
Encoding	MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQRFH_NONE	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQRFH_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQRFH MyRFH = {MQRFH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   StrucLength;     /* Total length of MQRFH including
                               NameValueString */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               NameValueString */
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows NameValueString */
  MQCHAR8  Format;          /* Format name of data that follows
                               NameValueString */
  MQLONG   Flags;           /* Flags */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQRFH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH including
                                     NameValueString */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows NameValueString */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows NameValueString */
  3 Format         char(8),       /* Format name of data that follows
                                     NameValueString */
  3 Flags          fixed bin(31); /* Flags */
```

Deklaration in High Level Assembler

```
MQRFH                DSECT
MQRFH_STRUCID        DS   CL4  Structure identifier
MQRFH_VERSION        DS   F    Structure version number
MQRFH_STRUCLENGTH    DS   F    Total length of MQRFH including
*                              NAMEVALUESTRING
MQRFH_ENCODING       DS   F    Numeric encoding of data that follows
*                              NAMEVALUESTRING
MQRFH_CODEDCHARSETID DS   F    Character set identifier of data that
*                              follows NAMEVALUESTRING
MQRFH_FORMAT         DS   CL8  Format name of data that follows
*                              NAMEVALUESTRING
MQRFH_FLAGS          DS   F    Flags
*
MQRFH_LENGTH         EQU  *-MQRFH
                     ORG  MQRFH
MQRFH_AREA           DS   CL(MQRFH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQRFH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH including'
                             'NameValueString'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'NameValueString'
  CodedCharSetId As Long     'Character set identifier of data that'
                             'follows NameValueString'
  Format         As String*8 'Format name of data that follows'
                             'NameValueString'
  Flags          As Long     'Flags'
End Type
```

MQRFH2 - Header 2 für Regeln und Formatierung

Dieser Abschnitt beschreibt den Regel- und Formatierungsheader der Version 2, welche Felder er enthält und die Anfangswerte dieser Felder.

Überblick zu MQRFH2

Verfügbarkeit

Alle WebSphere MQ-Systeme sowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Verwendungszweck

MQRFH2 basiert auf MQRFH, ermöglicht jedoch den Transport von Unicode-Zeichenfolgen ohne Konvertierung und die Übertragung numerischer Datentypen.

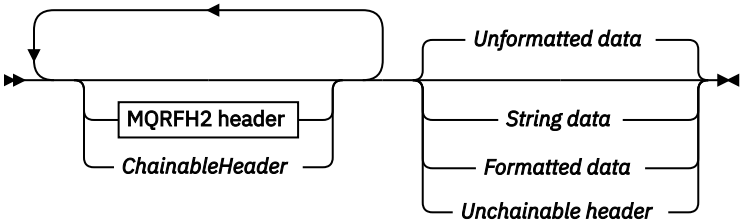
Die MQRFH2-Struktur definiert das Format des Regel- und Formatierungsheaders der Version 2. Verwenden Sie diesen Header, um Daten zu senden, die mit einer XML-ähnlichen Syntax codiert wurden. Eine Nachricht kann zwei oder mehrere MQRFH2-Strukturen in einer Reihe enthalten, wobei Benutzerdaten optional auf die letzte MQRFH2-Struktur in der Reihe folgen können.

Formatbezeichnung

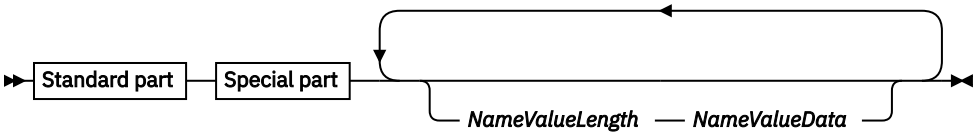
MQFMT_RF_HEADER_2

Syntax

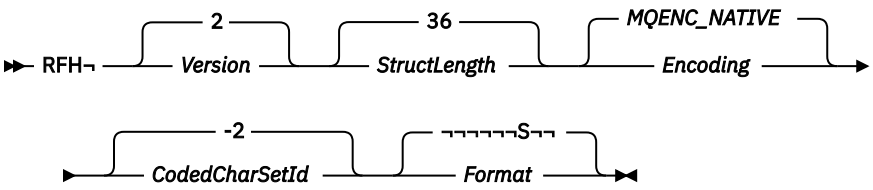
WebSphere MQ Message



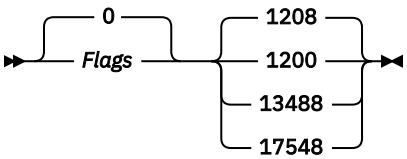
MQRFH2 header



Standard part



Special part



Zeichensatz und Codierung

Für den Zeichensatz, der bei der MQRFH2-Struktur verwendet wird, gelten spezielle Regeln:

- Zeichensatz und Codierung der Felder (außers *NameValueData*) entsprechen denen, die durch die Felder *CodedCharSetId* und *Encoding* in der Headerstruktur, die dem MQRFH2-Header vorausgeht, oder durch die gleichen Felder in der MQMD-Struktur definiert sind, falls sich der MQRFH2-Header am Anfang der Anwendungsnachrichtendaten befindet.

  Um in Warteschlangennamen gültige Zeichen bereitstellen zu können, muss der Zeichensatz Einzelbytezeichen unterstützen.

  Wenn MQGMO_CONVERT beim MQGET-Aufruf angegeben wird, konvertiert der Warteschlangenmanager alle MQRFH2-Felder außer *NameValueData* in den angeforderten Zeichensatz und die angeforderte Codierung.
- Der Zeichensatz von *NameValueData* entspricht dem, der durch das Feld *NameValueCCSID* definiert ist. Nur die aufgelisteten Unicode-Zeichensätze sind für *NameValueCCSID* gültig. Weitere Informationen finden Sie in der Beschreibung von *NameValueCCSID*.

  Einige Zeichensätze haben eine Darstellung, die von der Codierung abhängig ist. Wenn *NameValueCCSID* einer dieser Zeichensätze ist, muss die Codierung von *NameValueData* die gleiche sein, wie die der anderen Felder von MQRFH2.

  Wenn MQGMO_CONVERT beim MQGET-Aufruf angegeben wird, konvertiert der Warteschlangenmanager *NameValueData* in die angeforderte Codierung, ändert aber den Zeichensatz nicht.

Felder für MQRFH2

Die MQRFH2-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)

Gibt die Zeichensatzkennung der Daten an, die auf das letzte *NameValueData* -Feld folgen. Dies gilt nicht für Zeichendaten in der MQRFH2 -Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Folgende Sonderwerte sind zulässig:

MQCCSI_INHERIT
Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

MQCCSI_INHERIT kann nicht verwendet werden, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

Der Anfangswert dieses Felds ist MQCCSI_INHERIT.

Encoding (MQLONG)

Gibt die numerische Codierung der Daten an, die auf das letzte *NameValueData* -Feld folgen. Dies gilt nicht für numerische Daten in der MQRFH2 -Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

Flags (MQLONG)

Der Anfangswert dieses Felds ist MQRFH_NONE. MQRFH_NONE muss angegeben werden.

MQRFH_NONE
Keine Flags.

MQRFH_INTERNAL
Der Header MQRFH2 enthält intern festgelegte Eigenschaften.

MQRFH_INTERNAL ist zur Nutzung durch den Warteschlangenmanager vorgesehen.

Die oberen 16 Bits (MQRFH_FLAGS_RESTRICTED_MASK) sind für Flags, die der Warteschlangenmanager festlegt, reserviert. Flags, die möglicherweise ein Benutzer festlegt, werden in den unteren 16 Bits definiert.

Format (MQCHAR8)

Gibt den Formatnamen der Daten an, die auf das letzte Feld *NameValueData* folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds entsprechen denen für das Feld *Format* in MQMD.

Der Anfangswert dieses Felds ist MQFMT_NONE.

NameValueCCSID (MQLONG)

Gibt die ID des codierten Zeichensatzes der Daten im Feld *NameValueData* an. Dieser Zeichensatz unterscheidet sich von dem für andere Zeichenfolgen in der MQRFH2-Struktur und kann sich vom Zeichensatz der Daten (falls vorhanden) unterscheiden, die nach dem letzten Feld *NameValueData* am Ende der Struktur folgen.

NameValueCCSID muss einen der folgenden Werte übernehmen:

CCSID	Bedeutet
1200	UCS-2 uneingeschränkt
13488	UCS-2 Subset 2.0
17584	UCS-2 Subset 2.1 (enthält auch das Euro-Symbol)
1208	UTF-8

Bei UCS-2-Zeichensätzen muss die Codierung (Byteanordnung) des Feldes *NameValueData* mit der Codierung der anderen Felder in der MQRFH2-Struktur identisch sein. Ersatzzeichen (X'D800' bis X'DFFF') werden nicht unterstützt.

Anmerkung: Wenn *NameValueCCSID* keinen der oben aufgelisteten Werte hat und die MQRFH2-Struktur die Konvertierung beim MQGET-Aufruf erfordert, schließt der Aufruf mit dem Ursachencode MQRC_SOURCE_CCSID_ERROR ab und die Nachricht wird unkonvertiert zurückgegeben.

Der Anfangswert dieses Felds ist 1208.

NameValueData (MQCHARn)
NameValueData ist ein Feld variabler Länge, das einen Ordner mit Name/Wert-Paaren von Nachrichteneigenschaften enthält. Ein Ordner ist eine Zeichenfolge variabler Länge, der codierte Daten in einer XML-ähnlichen Syntax enthält. Die Länge der Zeichenfolge in Byte wird durch das Feld *NameValueLength* angegeben, das dem Feld *NameValueData* vorausgeht. Die Länge muss ein Vielfaches von vier sein.

Die Felder *NameValueLength* und *NameValueData* sind optional, aber wenn sie vorhanden sind, müssen sie als Paar auftreten und benachbart sein. Das Feldpaar kann so oft wie erforderlich wiederholt werden; Beispiel:

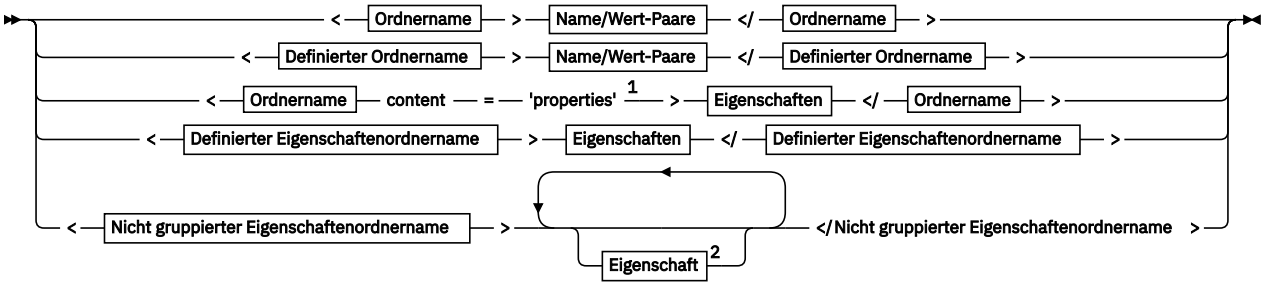
```
length1 data1 length2 data2 length3 data3
```

NameValueData wird *nicht* in den Zeichensatz konvertiert, der im Aufruf MQGET angegeben ist. Selbst wenn die Nachricht mit der Option MQGMO_CONVERT abgerufen wird, bleibt *NameValueData* im ursprünglichen Zeichensatz erhalten. Allerdings *wird NameValueData* in die Codierung konvertiert, die im Aufruf MQGET angegeben ist.

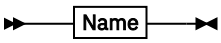
Anmerkung: Da es sich um optionale Felder handelt, fehlen sie in den Deklarationen der Struktur, die für die verschiedenen unterstützten Programmiersprachen bereitgestellt werden.

Anmerkung: Im Syntaxdiagramm werden die Begriffe "definiert" und "reserviert" verwendet. "Definiert" bedeutet, dass der Name von IBM WebSphere MQ verwendet wird. "Reserviert" bedeutet, dass der Name für eine künftige Verwendung durch WebSphere MQ reserviert ist.

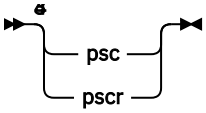
Syntax von *NameValueData*



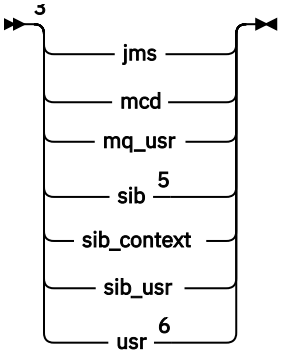
Ordnername



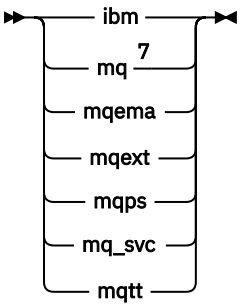
Definierter Ordnername



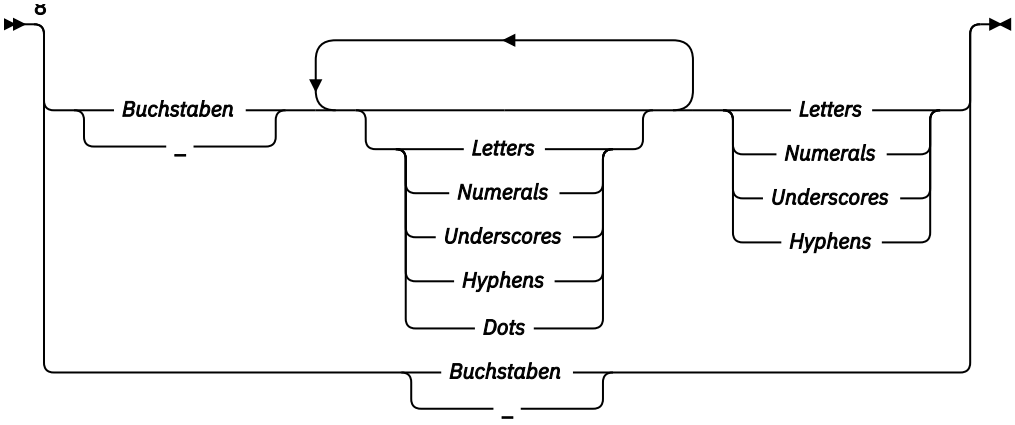
Definierter Eigenschaftenordnername



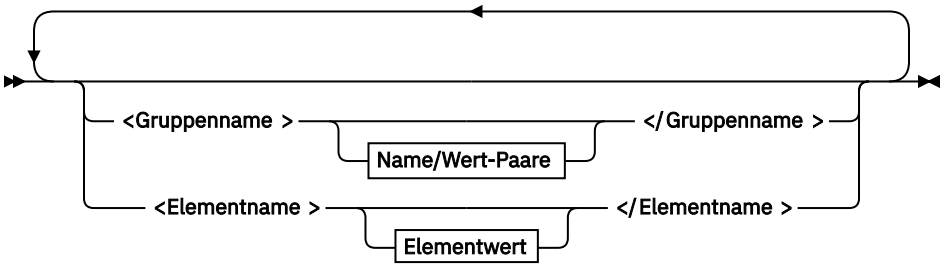
Nicht gruppierter Eigenschaftenordnername



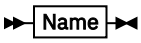
Ihren Namen



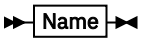
Name/Wert-Paare



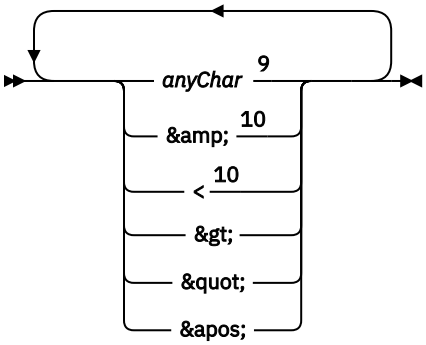
Gruppenname



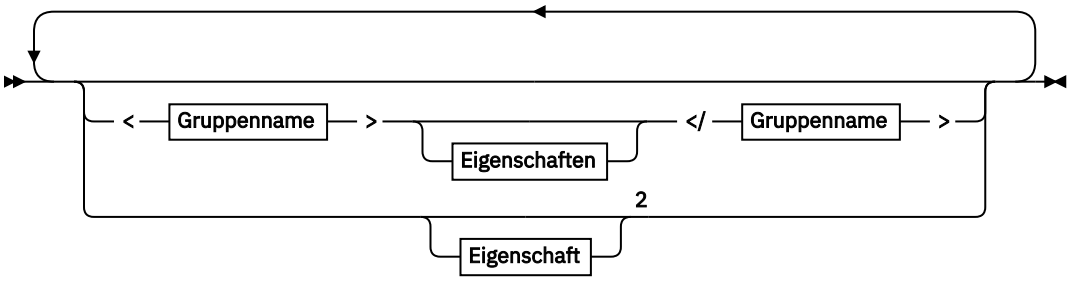
Elementname



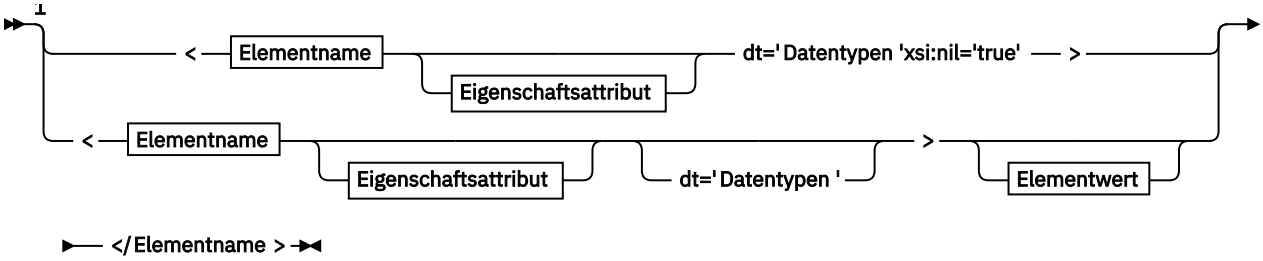
Elementwert



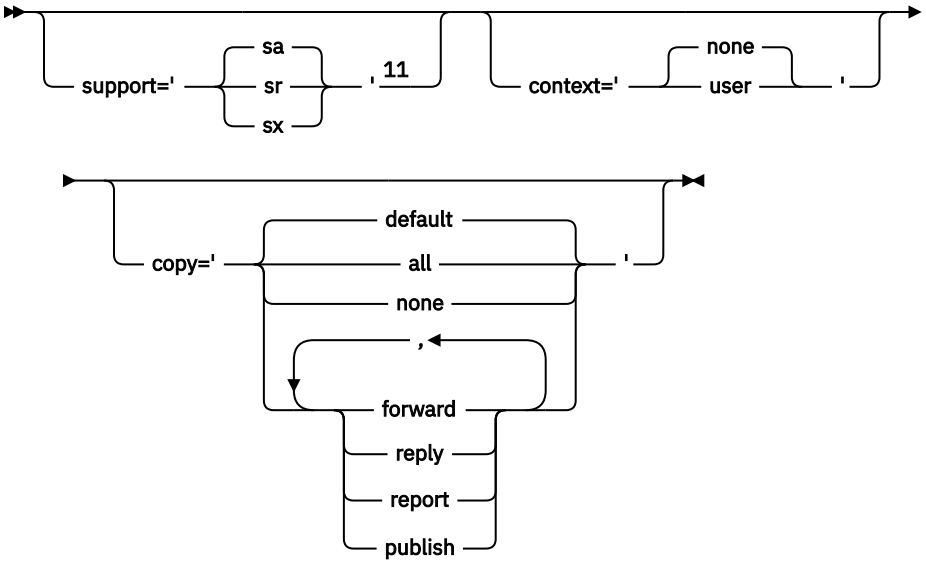
Eigenschaften



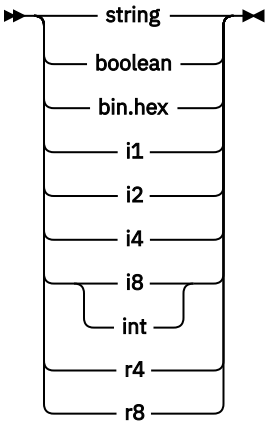
Eigenschaft



Eigenschaftsattribut



Datentypen



Anmerkungen:

[1] Anführungszeichen oder einfache Anführungszeichen sind gültig.

[2] Verwenden Sie keinen ungültigen Eigenschaftsnamen (siehe „Ungültiger Eigenschaftsname” auf Seite 531). Verwenden Sie einen reservierten Eigenschaftsnamen nur für seinen definierten Zweck (siehe „Definierte Eigenschaftsnamen” auf Seite 531).

[3] Der Name muss in Kleinbuchstaben geschrieben sein.

[4] Es wird nur ein einziger `psc`- und `pscr`-Ordner unterstützt.

[5] Nur Eigenschaften im ersten MQRFH2-Header sind bedeutsam. WebSphere Application Server Service Integration Bus ignoriert sib-, sib_context- und sib_usr-Ordner in nachfolgenden MQRFH2-Headern.

[6] Es darf nur ein `usr`-Ordner in einem MQRFH2-Header vorhanden sein. Eigenschaften im `usr`-Ordner dürfen nur einmal vorkommen.

[7] Nur Eigenschaften im ersten `mq`-Ordner sind bedeutsam. Wenn es sich um einen UTF-8-Ordner handelt, werden nur UTF-8-Einzelbytezeichen unterstützt. Das einzige Leerzeichen ist das Unicode-Zeichen `U+0020`.

[8] Gültige Zeichen sind in der W3C -XML-Spezifikation definiert und bestehen im Wesentlichen aus Unicode-Kategorien `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` und `Nd`.

[9] Alle Zeichen sind bedeutsam. Führende und abschließende Leerzeichen sind Teil des Elementwerts.

[10] Verwenden Sie kein ungültiges Zeichen (siehe „Ungültige Zeichen” auf Seite 531). Verwenden Sie eine Escapezeichenfolge anstelle dieser ungültigen Zeichen.

[11] Das Eigenschaftsattribut 'support' ist nur für den Ordner `mq` gültig.

Ordnername

NameValueData enthält einen einzelnen Ordner. Um mehrere Ordner zu erstellen, erstellen Sie mehrere *NameValueData*-Felder. Sie können mehrere *NameValueData*-Felder in einem einzelnen MQRFH2-Header innerhalb einer Nachricht erstellen. Alternativ können Sie mehrere verkettete MQRFH2-Header erstellen, die jeweils mehrere *NameValueData*-Felder enthalten.

Die Reihenfolge der MQRFH2-Header und die Reihenfolge der *NameValueData*-Felder hat keine Auswirkung auf den logischen Inhalt eines Ordners. Wenn derselbe Ordner mehrfach in einer Nachricht vorkommt, wird er als Ganzes syntaktisch analysiert. Wenn dieselbe Eigenschaft in mehreren Instanzen desselben Ordners vorkommt, wird sie als Liste syntaktisch analysiert.

Eine korrekte Syntaxanalyse von MQRFH2 wird nicht durch die alternativen Möglichkeiten beeinflusst, wie ein Ordner physisch in einer Nachricht gespeichert werden kann.

Für vier Ordner gilt diese Regel nicht. Nur die erste Instanz der Ordner `mq`, `sib`, `sib_context` und `sib_usr` wird syntaktisch analysiert.

Wenn dieselbe Eigenschaft mehrmals im kombinierten Inhalt der verketteten MQRFH2-Header auftritt, wird nur die erste Instanz der Eigenschaft geparst. Wenn eine Eigenschaft mit einem API-Aufruf festgelegt wird, z. B. mit MQSETMP, und direkt von einer Anwendung zu einem MQRFH2-Header hinzugefügt wird, hat der API-Aufruf Vorrang.

Ein Ordnername ist der Name eines Ordners, der Name/Wert-Paare oder Gruppen enthält. Gruppen und Name/Wert-Paar können auf derselben Ebene in der Ordnerbaumstruktur gemischt vorkommen (siehe Abbildung 1 auf Seite 520). Gruppennamen und Elementnamen dürfen nicht zusammen verwendet werden (siehe Abbildung 2 auf Seite 520).

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Abbildung 1. Richtige Verwendungen von Gruppen und Name/Wert-Paaren

```
<group1><nvp1>value</nvp1>value</group1>
```

Abbildung 2. Falsche Verwendung von Gruppen und Name/Wert-Paaren

Verwenden Sie keinen ungültigen oder reservierten Ordnernamen; siehe „Ungültiger Pfadname” auf Seite 530 und „Reservierter Ordner- oder Eigenschaftenordnername” auf Seite 530. Verwenden Sie einen definierten Ordnernamen nur für seinen definierten Zweck (siehe Abschnitt „Definierter Ordnername” auf Seite 521).

Wenn Sie das Attribut `'content=properties'` zum Ordnernamen-Tag hinzufügen, wird der Ordner zu einem Eigenschaftsordner (siehe Abbildung 3 auf Seite 521).

```
<myFolder></myfolder>
<myPropertyFolder contents='properties'></myPropertyFolder>
```

Abbildung 3. Beispiel eines Ordners und eines Eigenschaftenordners

Bei Ordnernamen muss die Groß-/Kleinschreibung beachtet werden. Für Ordnernamen und Eigenschaftenordnernamen gilt derselbe Namensbereich. Sie müssen unterschiedliche Namen haben. `Folder1` in Abbildung 4 auf Seite 521 muss ein anderer Name als `Folder2` in Abbildung 5 auf Seite 521sein.

```
<Folder1><NVP1>value</NVP1></Folder1>
```

Abbildung 4. `Folder1`-Namensbereich

```
<Folder2 content='properties'><Property1>value</Property1></Folder2>
```

Abbildung 5. `Folder2`-Namensbereich

Für Gruppen, Eigenschaften und Name/Wert-Paare in verschiedenen Ordnern gelten unterschiedliche Namensbereiche. `Property1` in Abbildung 5 auf Seite 521 ist eine andere Eigenschaft als `Property1` in Abbildung 6 auf Seite 521.

```
<Folder3 content='properties'><Property1>value</Property1></Folder3>
```

Abbildung 6. `Folder3`-Namensbereich

Eigenschaftenordner unterscheiden sich in zwei wichtigen Aspekten von Nicht-Eigenschaftenordnern:

1. Eigenschaftenordner enthalten Eigenschaften und Nicht-Eigenschaftenordner enthalten Name/Wert-Paare. Die Ordner unterscheiden sich leicht in syntaktischer Hinsicht.
2. Greifen Sie über die definierten Schnittstellen, z. B. die Eigenschaften-MQI oder JMS-Nachrichteneigenschaften, auf Nachrichteneigenschaften zu. Die Schnittstellen stellen sicher, dass die Eigenschaftenordner im `MQRFH2`-Header korrekt formatiert sind. Ein korrekt formatierter Eigenschaftenordner ist interoperabel zwischen Warteschlangenmanagern auf verschiedenen Plattformen und unter verschiedenen Releases.

Die MQI der Nachrichteneigenschaft ist eine leistungsfähige Methode zum Lesen und Schreiben einer `MQRFH2` und vermeidet die Schwierigkeiten, eine `MQRFH2` ordnungsgemäß zu parsen.

Definierter Ordnername

Ein definierter Ordnername ist der Name eines Ordners, der für die Verwendung durch WebSphere MQ oder ein anderes Produkt reserviert ist. Erstellen Sie keinen Ordner mit demselben Namen und fügen Sie Ihre Name/Wert-Paare nicht zu den Ordnern hinzu. Die definierten Ordner sind `psc` und `pscr`.

`psc` und `pscr` werden von eingereihtem Publish/Subscribe verwendet.

Eine segmentierte Nachricht, die mit MQMF_SEGMENT oder MQMF_SEGMENTATION_ALLOWED eingereiht wird, darf keinen MQRFH2 mit einem definierten Ordnernamen enthalten. Der Aufruf MQPUT schlägt mit Ursachencode 2443 (MQRC_SEGMENTATION_NOT_ALLOWED) fehl.

Definierter Eigenschaftenordnername

Ein definierter Eigenschaftenordnername ist der Name eines Eigenschaftenordners, der für die Verwendung durch IBM WebSphere MQ oder ein anderes Produkt reserviert ist. Informationen zu den Namen der Ordner und deren Inhalten finden Sie im Abschnitt Eigenschaftenordner. Definierte Eigenschaftenordnernamen sind eine Untergruppe aller Ordnernamen, die von WebSphere MQ reserviert sind (siehe „Reservierter Ordner- oder Eigenschaftenordnername” auf Seite 530).

Ein in einem definierten Eigenschaftenordner gespeichertes Element ist eine Eigenschaft. Ein in einem definierten Eigenschaftsordner gespeichertes Element darf kein Attribut content='properties' aufweisen.

Sie können Eigenschaften nur zu den definierten Eigenschaftsordnern usr, mq_usrund sib_usrhinzufügen. In anderen Eigenschaftsordnern, wie mq und sib, ignoriert WebSphere MQ Eigenschaften, die nicht erkannt werden, oder löst sie aus.

In der Beschreibung jedes definierten Eigenschaftenordners werden die Eigenschaften aufgelistet, die von IBM WebSphere MQ definiert wurden und von Anwendungsprogrammen verwendet werden können. Auf einige der Eigenschaften wird indirekt durch Festlegen oder Abrufen einer JMS-Eigenschaft und auf andere direkt mit den MQI-Aufrufen MQSETMP und MQINQMP zugegriffen.

Die definierten Eigenschaftenordner enthalten zudem andere Eigenschaften, die von IBM WebSphere MQ reserviert wurden, auf die von Anwendungen jedoch nicht zugegriffen werden kann. Die Namen der reservierten Eigenschaften werden nicht aufgelistet. In den Eigenschaftsordnern usr, mq_usrund sib_usr sind keine reservierten Eigenschaften vorhanden. Aber erstellen Sie keine Eigenschaften mit ungültigen Eigenschaftsnamen (siehe „Ungültiger Eigenschaftsname” auf Seite 531).

Eigenschaftenordner

jms

jms enthält JMS-Headerfelder und JMSX-Eigenschaften, die nicht vollständig in MQMD ausgedrückt werden können. Der Ordner jms ist immer in einem JMS-MQRFH2-Header enthalten.

Tabelle 532. jms-Eigenschaftsname, -Synonym, -Datentyp und -Ordner

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
JMSDestination	jms.Dst	string	<jms><Dst>*destination*</Dst></jms>
JMSExpiration	jms.Exp	i8	<jms><Exp>*expiration*</Exp></jms>
JMSCorrelation	jms.Cid	string	<jms><Cid>*correlationId*</Cid></jms>
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>

Tabelle 532. jms-Eigenschaftsname, -Synonym, -Datentyp und -Ordner (Forts.)			
Eigenschaftssynonym	**Eigenschaftsname**	**Datentyp**	**Ordner**
JMSXGroupID	jms.Gid	string	<jms><Gid>*groupId*</Gid></jms>
JMSXGroupSeq	jms.Seq	i4	<jms><Seq>*messageSequenceNo*</Seq></jms>

Fügen Sie keine eigenen Eigenschaften zum Ordner jms hinzu.

mcd

mcd enthält Eigenschaften, die das Format der Nachricht beschreiben. Beispielsweise identifiziert die Eigenschaft Msd (Message Service Domäne = Nachrichtenservicedomäne) eine JMS-Nachricht als JMSTextMessage, JMSBytesMessage, JMSStreamMessage, JMSMapMessage, JMSObjectMessage oder 0.

Der Ordner mcd ist immer in einer JMS-Nachricht mit MQRFH2 enthalten.

Er ist immer in einer Nachricht mit einem MQRFH2-Header enthalten, die vom WebSphere Message Broker gesendet wurde. Er beschreibt die Domäne, das Format, den Typ und die Nachrichtengruppe einer Nachricht.

Tabelle 533. mcd-Eigenschaftsname, -Synonym, -Datentyp und -Ordner			
Eigenschaftssynonym	**Eigenschaftsname**	**Datentyp**	**Ordner**
	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>

Fügen Sie keine eigenen Eigenschaften zum Ordner mcd hinzu.

mq_usr

mq_usr enthält anwendungsdefinierte Eigenschaften, die nicht als benutzerdefinierte JMS-Eigenschaften verfügbar sind. In diesem Ordner können Eigenschaften, die JMS-Anforderungen nicht erfüllen, abgelegt werden.

Sie können Eigenschaften im Ordner mq_usr erstellen. Eigenschaften, die Sie im mq_usr erstellen, ähneln Eigenschaften, die Sie in neuen Ordnern mit dem Attribut content='properties' erstellen.

sib

sib enthält WebSphere Application Server Service Integration Bus (WAS/SIB)-Systemnachrichteneigenschaften. sib -Eigenschaften werden nicht als JMS-Eigenschaften für IBM WebSphere MQ -JMS-Anwendungen bereitgestellt, weil sie nicht den unterstützten Typ aufweisen. Beispielsweise können einige sib-Eigenschaften deshalb nicht als JMS-Eigenschaften bereitgestellt werden, weil es sich um Bytefeldgruppen handelt. Einige sib-Eigenschaften sind für WAS/SIB-Anwendungen als JMS_IBM_*-Eigenschaften verfügbar; dazu gehören Eigenschaften für Vorwärts- und Rückwärts-Routing-Pfade.

Fügen Sie keine eigenen Eigenschaften zum Ordner sib hinzu.

sib_context

sib_context enthält WAS/SIB-Systemnachrichteneigenschaften, die nicht für WAS/SIB-Benutzeranwendungen oder als JMS-Eigenschaften verfügbar sind. sib_context enthält Sicherheits- und Transaktionseigenschaften, die für Web-Services verwendet werden.

Fügen Sie keine eigenen Eigenschaften zum Ordner sib_context hinzu.

sib_usr

sib_usr enthält WAS/SIB-Benutzernachrichteneigenschaften, die nicht als JMS-Benutzereigenschaften verfügbar sind, da sie nicht zu den unterstützten Typen gehören. sib_usr ist für WAS/SIB-Anwendungen in der Schnittstelle SIMessage verfügbar (siehe Serviceintegration entwickeln).

Der Typ einer sib_usr-Eigenschaft muss bin.hex sein und der Wert muss das richtige Format aufweisen. Wenn eine IBM WebSphere MQ-Anwendung ein Element des Typs bin.hex in einem falschen Format in den Ordner schreibt, wird eine IOException-Ausnahme an die Anwendung zurückgegeben. Wenn die Eigenschaft nicht den Typ bin.hex hat, wird eine ClassCastException-Ausnahme an die Anwendung zurückgegeben.

Versuchen Sie nicht, über diesen Ordner JMS-Benutzereigenschaften für WAS/SIB bereitzustellen; verwenden Sie stattdessen den Ordner usr.

Sie können Eigenschaften im Ordner sib_usr erstellen.

usr

usr enthält anwendungsdefinierte JMS-Eigenschaften, die der Nachricht zugeordnet sind. Der Ordner usr ist nur dann vorhanden, wenn eine Anwendung eine anwendungsdefinierte Eigenschaft festgelegt hat.

usr ist der Standardeigenschaftenordner. Wenn eine Eigenschaft ohne einen Ordnernamen festgelegt wird, wird sie im Ordner usr abgelegt.

Tabelle 534. usr-Eigenschaftsname, -Synonym, -Datentyp und -Ordner.

Die Web-Services-Eigenschaftswerte werden im Abschnitt MQRFH2-SOAP-Einstellungen beschrieben.

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
	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>

Sie können Eigenschaften im Ordner usr erstellen.

Eine segmentierte Nachricht, die mit MQMF_SEGMENT oder MQMF_SEGMENTATION_ALLOWED eingereiht wird, darf keinen MQRFH2 mit einem definierten Eigenschaftsordnernamen enthalten. Der Aufruf MQPUT schlägt mit Ursachencode 2443 (MQRC_SEGMENTATION_NOT_ALLOWED) fehl.

Nicht gruppierter Eigenschaftenordnername

ibm

ibm enthält Eigenschaften, die nur von IBM WebSphere MQ verwendet werden.

Tabelle 535. ibm-Eigenschaftsname, -Synonym, -Datentyp und -Ordner

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

Fügen Sie keine eigenen Eigenschaften zum Ordner ibm hinzu.

mq

mq enthält Eigenschaften, die nur von IBM WebSphere MQ verwendet werden.

Die folgenden Einschränkungen gelten für Eigenschaften im Ordner mq :

- Nur Eigenschaften im ersten wichtigen Ordner mq in der Nachricht werden von MQbearbeitet; Eigenschaften in allen anderen mq -Ordnern in der Nachricht werden ignoriert.
- Der Ordner darf nur UTF-8-Einzelbytezeichen enthalten. Ein Mehrfachbytezeichen im Ordner kann dazu führen, dass die Syntaxanalyse fehlschlägt und die Nachricht abgelehnt wird.
- Im Ordner sollten keine Escapezeichenfolgen verwendet werden. Eine Escapezeichenfolge wird wie der eigentliche Wert des Elements behandelt.
- Nur Unicode-Zeichen U+0020 werden als Leerzeichen innerhalb des Ordners behandelt. Alle anderen Zeichen werden als bedeutsam betrachtet und können dazu führen, dass die Syntaxanalyse fehlschlägt und die Nachricht abgelehnt wird.

Wenn das Parsing des Ordners mq fehlschlägt oder wenn der Ordner diese Einschränkungen nicht beachtet, wird die Nachricht mit dem Ursachencode 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR abgelehnt.

Fügen Sie keine eigenen Eigenschaften zum Ordner mq hinzu.

mqema

mqema enthält Eigenschaften, die nur von WebSphere Application Server verwendet werden. Der Ordner wurde durch mqext ersetzt.

Fügen Sie keine eigenen Eigenschaften zum Ordner mqema hinzu.

mqext

mqext enthält Eigenschaften, die nur von WebSphere Application Server verwendet werden. Der Ordner ist nur vorhanden, wenn die Anwendung mindestens eine der von IBM definierten Eigenschaften festgelegt hat.

Tabelle 536. mqext-Eigenschaftsname, -Synonym, -Datentyp und -Ordner

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>

Fügen Sie keine eigenen Eigenschaften zum Ordner mqext hinzu.

mqps

mqps enthält Eigenschaften, die nur von IBM WebSphere MQ Publish/Subscribe verwendet werden. Der Ordner ist nur vorhanden, wenn die Anwendung mindestens eine der integrierten Publish/Subscribe-Eigenschaften festgelegt hat.

Tabelle 537. mqps-Eigenschaftsname, -Synonym, -Datentyp und -Ordner

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
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>

Fügen Sie keine eigenen Eigenschaften zum Ordner mqps hinzu.

mq_svc

mq_svc enthält Eigenschaften, die von SupportPac verwendet werden MA93.

Fügen Sie keine eigenen Eigenschaften zum Ordner mq_svc hinzu.

mqtt

mqtt enthält Eigenschaften, die nur von IBM WebSphere MQ Telemetry verwendet werden.

Tabelle 538. mqtt-Eigenschaftsname, -Synonym, -Datentyp und -Ordner

Eigenschaftssynonym	Eigenschaftsname	Datentyp	Ordner
	mqtt.clientId	string	<mqtt><clientId>*topicString*</clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos>*qualityOfService*</qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid>*messageIdentifier*</msgid></mqtt>

Fügen Sie keine eigenen Eigenschaften zum Ordner mqtt hinzu.

Eine segmentierte Nachrichteneinreihung mit MQMF_SEGMENT oder MQMF_SEGMENTATION_ALLOWED darf keinen MQRFH2 mit einem nicht gruppierten Eigenschaftsordnernamen enthalten. Der Aufruf MQPUT schlägt mit Ursachencode 2443 (MQRC_SEGMENTATION_NOT_ALLOWED) fehl.

Name/Wert-Paare

Im Syntaxdiagramm beschreiben "Name/Wert-Paare" den Inhalt eines gewöhnlichen Ordners. Ein gewöhnlicher Ordner enthält Gruppen und Elemente. Ein Element ist ein Name/Wert-Paar. Eine Gruppe enthält Elemente und andere Gruppen.

Übertragen auf Baumstrukturen, sind Elemente Blattknoten und Gruppen interne Knoten. Ein interner Knoten und der Ordner, bei dem es sich um den Stammknoten handelt, können eine Mischung aus internen Knoten und Blattknoten enthalten. Ein Knoten kann nicht gleichzeitig ein interner Knoten und ein Blattknoten sein (siehe Abbildung 2 auf Seite 520).

Eigenschaften

Im Syntaxdiagramm beschreiben "Eigenschaften" den Inhalt eines Eigenschaftenordners. Ein Eigenschaftenordner enthält Gruppen und Eigenschaften. Eine Eigenschaft ist ein Name/Wert-Paar mit einem optionalen Datentypattribut. Eine Gruppe enthält Eigenschaften und andere Gruppen.

Übertragen auf Baumstrukturen, sind Eigenschaften Blattknoten und Gruppen interne Knoten. Ein interner Knoten und der Eigenschaftenordner, bei dem es sich um den Stammknoten handelt, können eine Mischung aus internen Knoten und Blattknoten enthalten. Ein Knoten kann nicht gleichzeitig ein interner Knoten und ein Blattknoten sein (siehe Abbildung 2 auf Seite 520).

Eigenschaft

Eine Nachrichteneigenschaft ist ein Name/Wert-Paar in einem Eigenschaftenordner. Sie kann optional ein Datentypattribut und ein Eigenschaftsattribut enthalten (Beispiel siehe Abbildung 7 auf Seite 527). Wenn das Datentypattribut nicht angegeben ist, ist der Eigenschaftstyp `string`.

```
<pf><p1 dt='i8' >value</p1></pf>
```

Abbildung 7. Datentypattribut

Der Name einer Nachrichteneigenschaft ist der vollständige Pfadname, wobei die XML-ähnliche <>-Syntax durch Punkte ersetzt wird. Beispielsweise wird `myPropertyFolder1.myGroup1.myGroup2.myProperty1` einer *NameValueData*-Zeichenfolge in Abbildung 8 auf Seite 527 zugeordnet. Die Zeichenfolge ist für einfacheres Lesen formatiert.

```
<myPropertyFolder1>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
        </myGroup2>
    </myGroup1>
</myPropertyFolder1>
```

Abbildung 8. Zuordnung eines einzelnen Eigenschaftsnamens

Ein Eigenschaftenordner kann mehrere Eigenschaften enthalten. Beispiel: Die Eigenschaften in Abbildung 9 auf Seite 528 werden dem Eigenschaftenordner in Abbildung 10 auf Seite 528 zugeordnet.

```
myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Abbildung 9. Mehrere Eigenschaften mit demselben Stammnamen

```
<myPropertyFolder1>
    <myProperty4>value</myProperty4>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
            <myProperty2>value</myProperty2>
        </myGroup2>
        <myProperty3>value</myProperty3>
    </myGroup1>
</myPropertyFolder1>
```

Abbildung 10. Zuordnung mehrerer Eigenschaftsnamen

Ihren Namen

Ein Name muss mit einem *Buchstaben* oder einem *Unterstreichungszeichen* beginnen. Er darf keinen *Doppelpunkt* enthalten, nicht mit einem *Punkt* enden und nur *Letters*, *Numerals*, *Underscores*, *Hyphens* und *Dots* enthalten.Gültige Zeichen sind in der W3C -XML-Spezifikation definiert und bestehen im Wesentlichen aus Unicode-Kategorien `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` und `Nd`.

Der vollständige Pfad einer Eigenschaft oder eines Name/Wert-Paares darf nicht gegen die im Abschnitt „Ungültiger Pfadname” auf Seite 530 beschriebene Regel verstoßen. Pfade sind auf 4095 Byte beschränkt, dürfen keine Unicode-Kompatibilitätszeichen enthalten und dürfen nicht mit der Zeichenfolge XML beginnen.

Gruppenname

Ein Gruppenname hat dieselbe Syntax wie ein Name. Die Angabe von Gruppennamen ist optional. Eigenschaften und Name/Wert-Paare können im Stamm eines Ordners abgelegt werden. Verwenden Sie Gruppen, wenn es für die Verwaltung von Eigenschaften und Name/Wert-Paaren hilfreich ist.

Elementname

Ein Elementname hat dieselbe Syntax wie ein Name.

Elementwert

Ein Elementwert enthält alle Leerzeichen zwischen dem Tag <*Element name*> und </*Element name*>. Verwenden Sie nicht die beiden Zeichen < und & in einem Wert. Ersetzen Sie sie anschließend durch < und `&`.

Eigenschaftsattribut

Über die Eigenschaftsattribute werden Eigenschaftendeskriptorfelder zugeordnet. Es bestehen folgende Zuordnungen:

Support

sa
MQPD_SUPPORT_OPTIONAL

sr
MQPD_SUPPORT_REQUIRED

sx
MQPD_SUPPORT_REQUIRED_IF_LOCAL

Context

none
MQPD_NO_CONTEXT

user
MQPD_USER _CONTEXT

CopyOptions

forward
MQPD_COPY_FORWARD

reply
MQPD_COPY_REPLY

report
MQPD_COPY_REPORT

publish
MQPD_COPY_PUBLISH

all
MQPD_COPY_ALL

Verwenden Sie `all` nicht in Kombination mit anderen Optionen.

default
MQPD_COPY_DEFAULT

Verwenden Sie `default` nicht in Kombination mit anderen Optionen. `default` entspricht `forward` + `report` + `publish`

none
MQPD_COPY_NONE

Verwenden Sie none nicht in Kombination mit anderen Optionen.

Die Unterstützung -Eigenschaftsattribute gelten nur für Eigenschaften im Ordner `mq` .

Die Context- und CopyOptions-Eigenschaftsattribute sind auf alle Eigenschaftenordner anwendbar.

Datentyp

MQRFH2-Datentypen werden Nachrichteneigenschaftstypen wie folgt zugeordnet:

Tabelle 539. Datentypzuordnungen

MQRFH2-Datentyp	**Nachrichteneigenschaftstyp**
bin.hex	MQBYTE[]
boolean	MQBOOL
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Bei Elementen ohne Datentyp wird der Typ `string` vorausgesetzt.

Ein Nullwert wird durch das Elementattribut `xsi:nil='true'` angegeben. Verwenden Sie das Attribut `xsi:nil='false'` nicht für Werte ungleich null. Folgende Eigenschaft hat beispielsweise einen Nullwert:

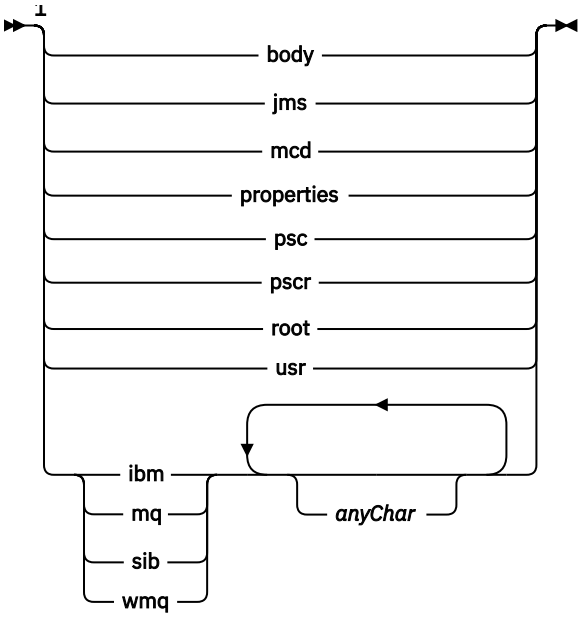
```
<NullProperty
xsi:nil='true'></NullProperty>
```

Eine Byte- oder Zeichenfolgeeigenschaft kann einen leeren Wert enthalten. Ein leerer Wert wird durch ein Element MQRFH2 mit einem Elementwert mit der Länge null dargestellt. Folgende Eigenschaft enthält beispielsweise einen leeren Wert:

```
<EmptyProperty></EmptyProperty>
```

Reservierter Ordner- oder Eigenschaftenordnername

Der Name eines Ordners oder Eigenschaftenordners darf nicht mit einer der folgenden Zeichenfolgen beginnen. Die Präfixe sind für Ordner oder Eigenschaftennamen reserviert, die von IBM erstellt werden.

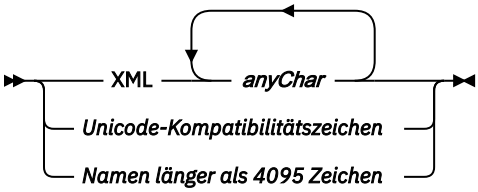


Anmerkungen:

[1] Ein reservierter Ordner- oder Eigenschaftsname besteht aus einer Mischung aus Klein- und Großbuchstaben.

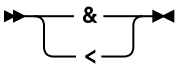
Ungültiger Pfadname

Der vollständige Pfad eines Name/Wert-Paares oder einer Eigenschaft darf keine der folgenden Zeichenfolgen enthalten.



Ungültige Zeichen

Verwenden Sie immer die Escape-Zeichenfolgen & und < anstelle der Literale "&" und "<".

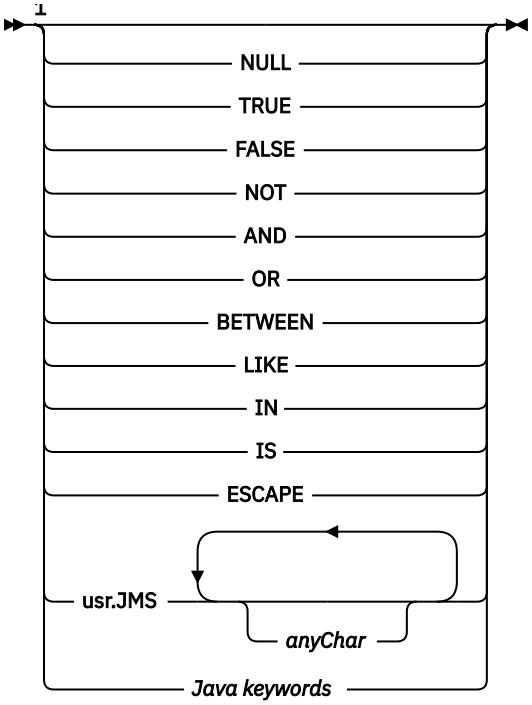


Definierte Eigenschaftsnamen

Definierte Eigenschaftsnamen sind die Namen von Eigenschaften, die von WebSphere MQ oder anderen Produkten definiert und von IBM WebSphere MQ und Benutzeranwendungen verwendet werden. Definierte Eigenschaften sind nur in definierten Eigenschaftenordnern enthalten. Definierte Eigenschaftsnamen werden in der Beschreibung von Eigenschaftenordnern beschrieben (siehe Eigenschaftenordner).

Ungültiger Eigenschaftsname

Erstellen Sie keine Eigenschaftsnamen, die der folgenden Regel entsprechen. Die Regel gilt für den vollständigen Eigenschaftspfad, der den Namen einer Eigenschaft bildet, und nicht nur für den Eigenschaftselementnamen.



Anmerkungen:

[1] Ein ungültiger Eigenschaftsname kann eine beliebige Kombination aus Klein- und Großbuchstaben enthalten.

NameValueLength (MQLONG)
Die Länge des entsprechenden Felds NameValueData.

Gibt die ID Länge in Byte der Daten im Feld *NameValueData* an. *NameValueLength* muss ein Vielfaches von vier sein.

Anmerkung: Die Felder *NameValueLength* und *NameValueData* sind optional, müssen aber, wenn Sie angegeben sind, als Paar auftreten und aufeinander folgen. Das Feldpaar kann so oft wie erforderlich wiederholt werden; Beispiel:

```
length1 data1 length2 data2 length3 data3
```

Da es sich um optionale Felder handelt, fehlen sie in den Deklarationen der Struktur, die für die verschiedenen unterstützten Programmiersprachen bereitgestellt werden.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQRFH_STRUC_ID
ID für Regel- und Formatierungsheaderstuktur.

Für die Programmiersprache C ist auch die Konstante MQRFH_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQRFH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQRFH_STRUC_ID.

StrucLength (MQLONG)

Dabei handelt es sich um die Länge der MQRFH2-Struktur in Bytes, einschließlich der Felder *NameValueLength* und *NameValueData* am Ende der Struktur. Es ist gültig, wenn es mehrere Paare der Felder *NameValueLength* und *NameValueData* am Ende der Struktur gibt, in folgender Sequenz:

```
length1, data1, length2, data2, ...
```

StrucLength umfasst *keine* Benutzerdaten, die auf das letzte Feld *NameValueData* am Ende der Struktur folgen.

Um Probleme bei der Konvertierung von Benutzerdaten in bestimmten Umgebungen zu vermeiden, muss *StrucLength* ein Vielfaches von vier sein.

Die folgende Konstante gibt die Länge der *Festkomponente* der Struktur an, d. h. die Länge ausschließlich der Felder *NameValueLength* und *NameValueData*:

MQRFH_STRUC_LENGTH_FIXED_2
Länge der Festkomponente der MQRFH2-Struktur.

Der Anfangswert dieses Felds ist MQRFH_STRUC_LENGTH_FIXED_2.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQRFH_VERSION_2
Regel- und Formatierungsheaderstuktur der Version 2.

Der Anfangswert dieses Felds ist MQRFH_VERSION_2.

Anfangswerte und Sprachendeklarationen für MQRFH2

Tabelle 540. Anfangswerte für Felder im MQRFH2

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQRFH_STRUC_ID	'RFH¬'
Version	MQRFH_VERSION_2	2
StrucLength	MQRFH_STRUC_LENGTH_FIXED_2	36
Encoding	MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	MQCCSI_INHERIT	-2
Format	MQFMT_NONE	Leerzeichen
Flags	MQRFH_NONE	0

Tabelle 540. Anfangswerte für Felder im MQRFH2 (Forts.)		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**
NameValueCCSID	--	1208

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQRFH2_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   StrucLength;     /* Total length of MQRFH2 including all
                               NameValueLength and NameValueData
                               fields */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               last NameValueData field */
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows last NameValueData field */
  MQCHAR8  Format;          /* Format name of data that follows last
                               NameValueData field */
  MQLONG   Flags;           /* Flags */
  MQLONG   NameValueCCSID;  /* Character set identifier of
                               NameValueData */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQRFH2 based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH2 including
                                     all NameValueLength and
                                     NameValueData fields */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
```

```
                                    follows last NameValueData field */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows last NameValueData
                                     field */
  3 Format         char(8),       /* Format name of data that follows
                                     last NameValueData field */
  3 Flags          fixed bin(31), /* Flags */
  3 NameValueCCSID fixed bin(31); /* Character set identifier of
                                     NameValueData */
```

Deklaration in High Level Assembler

```
MQRFH                 DSECT
MQRFH_STRUCID         DS   CL4  Structure identifier
MQRFH_VERSION         DS   F    Structure version number
MQRFH_STRUCLENGTH     DS   F    Total length of MQRFH2 including all
*                               NAMEVALUELENGTH and NAMEVALUEDATA fields
MQRFH_ENCODING        DS   F    Numeric encoding of data that follows
*                               last NAMEVALUEDATA field
MQRFH_CODEDCHARSETID  DS   F    Character set identifier of data that
*                               follows last NAMEVALUEDATA field
MQRFH_FORMAT          DS   CL8  Format name of data that follows last
*                               NAMEVALUEDATA field
MQRFH_FLAGS           DS   F    Flags
MQRFH_NAMEVALUECCSID  DS   F    Character set identifier of
*                               NAMEVALUEDATA
*
MQRFH_LENGTH          EQU  *-MQRFH
                      ORG  MQRFH
MQRFH_AREA            DS   CL(MQRFH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQRFH2
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH2 including all'
                             'NameValueLength and NameValueData fields'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'last NameValueData field'
  CodedCharSetId As Long     'Character set identifier of data that'
                             'follows last NameValueData field'
  Format         As String*8 'Format name of data that follows last'
                             'NameValueData field'
  Flags          As Long     'Flags'
  NameValueCCSID As Long     'Character set identifier of NameValueData'
End Type
```

MQRMH - Header für Referenznachrichten

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 541. Felder für MQRMH

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
StrucLength	Gesamtlänge von MQRMH, einschließlich Zeichenfolgen am Ende von festen Feldern, aber ohne Massendaten	StrucLength
Encoding	Numerische Codierung von Massendaten	Codierung
CodedCharSetId	Zeichensatzkennung von Massendaten	CodedCharSetId
Format	Name des Formats von Massendaten	Format

Tabelle 541. Felder für MQRMH (Forts.)		
Feld	**Beschreibung**	**Thema**
Flags	Referenznachrichtenflags	Flags
ObjectType	Objekttyp	ObjectType
ObjectInstanceId	Objektinstanz-ID	ObjectInstanceId
SrcEnvLength	Länge der Quellenumgebungsdaten	SrcEnvLength
SrcEnvOffset	Offset der Quellenumgebungsdaten	SrcEnvOffset
SrcNameLength	Länge des Quellenobjektnamens	SrcNameLength
SrcNameOffset	Offset des Quellenobjektnamens	SrcNameOffset
DestEnvLength	Länge der Zielumgebungsdaten	DestEnvLength
DestEnvOffset	Offset der Zielumgebungsdaten	DestEnvOffset
DestNameLength	Länge des Zielobjektnamens	DestNameLength
DestNameOffset	Offset des Zielobjektnamens	DestNameOffset
DataLogicalLength	Länge der Massendaten	DataLogicalLength
DataLogicalOffset	Unterer Offset der Massendaten	DataLogicalOffset
DataLogicalOffset2	Oberer Offset der Massendaten	DataLogicalOffset2

Überblick zu MQRMH

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQRMH-Struktur definiert das Format eines Referenznachrichtenheaders. Dieser Header wird mit benutzerdefinierten Nachrichtenkanalexits verwendet, um extrem große Datenvolumen (*Massendaten*) von einem Warteschlangenmanager zum nächsten zu senden. Der Unterschied zu normalem Messaging besteht darin, dass Massendaten nicht in einer Warteschlange gespeichert werden; es wird nur eine *Referenz* auf die Massendaten in der Warteschlange gespeichert. Dies verringert die Wahrscheinlichkeit, dass MQ-Ressourcen durch eine geringe Anzahl extrem umfangreicher Nachrichten erschöpft werden.

Formatname: MQFMT_REF_MSG_HEADER.

Zeichensatz und Codierung: Zeichendaten im MQRMH und die von den Offsetfeldern betroffenen Zeichenfolgen müssen dem Zeichensatz des lokalen Warteschlangenmanagers entsprechen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird. Numerische Daten im MQRMH müssen der nativen Systemcodierung entsprechen, die durch den Wert von MQENC_NATIVE für die Programmiersprache C angegeben wird.

Legen Sie den Zeichensatz und die Codierung von MQRM in den Feldern *CodedCharSetId* und *Encoding* wie folgt fest:

- in MQMD (wenn sich die MQRMH-Struktur am Anfang der Nachrichtendaten befindet) oder
- in der Headerstruktur, die der MQRMH-Struktur vorausgeht (alle anderen Fälle).

Verwendung: Eine Anwendung reiht eine Nachricht ein, die aus einem MQRMH besteht, nicht aber die Massendaten enthält. Wenn ein Nachrichtenkanalagent (MCA, Message Channel Agent) die Nachricht aus der Übertragungswarteschlange liest, wird ein benutzerdefinierter Nachrichtenexit aufgerufen, um den Referenznachrichtenheader zu verarbeiten. Der Exit kann an die Referenznachricht die durch die MQRMH-Struktur angegebenen Massendaten anhängen, bevor der MCA die Nachricht durch den Kanal an den nächsten Warteschlangenmanager weitersendet.

Auf der Empfangsseite muss ein Nachrichtenexit existieren, der auf die Referenznachrichten wartet. Wenn eine Referenznachricht empfangen wird, muss der Exit das Objekt aus den Massendaten erstellen,

die auf den MQRMH in der Nachricht folgen, und dann die Referenznachricht ohne die Massendaten weitergeben. Die Referenznachricht kann später durch eine Anwendung abgerufen werden, die die Referenznachricht (ohne die Massendaten) aus einer Warteschlange abliest.

Normalerweise steht in der Nachricht nur die MQRMH-Struktur. Wenn sich die Nachricht jedoch in einer Übertragungswarteschlange befindet, sind ein oder mehrere zusätzliche Header der MQRMH-Struktur vorangestellt.

Eine Referenznachricht kann auch an eine Verteilerliste gesendet werden. In diesem Fall gehen die MQDH-Struktur und die zugehörigen Datensätze der MQRMH-Struktur voran, wenn sich die Nachricht in einer Übertragungswarteschlange befindet.

Anmerkung: Senden Sie eine Referenznachricht nicht als segmentierte Nachricht, weil der Nachrichtenexit diese nicht ordnungsgemäß verarbeiten kann.

Datenkonvertierung: Bei der Datenkonvertierung umfasst das Konvertieren der MQRMH-Struktur eine Konvertierung der Quellenumgebungsdaten, des Quellenobjektnamens, der Zielumgebungsdaten und des Zielobjektnamens. Alle anderen Bytes innterhalb von *StrucLength* Bytes am Anfang der Struktur werden entweder verworfen oder haben nach der Datenkonvertierung nicht definierte Werte. Die Massendaten werden konvertiert, wenn alle folgenden Aussagen zutreffen:

- Die Massendaten sind in der Nachricht vorhanden, wenn die Datenkonvertierung durchgeführt wird.
- Das Feld *Format* im MQRMH hat einen anderen Wert als MQFMT_NONE.
- Ein benutzerdefinierter Datenkonvertierungsexit mit angegebenen Formatnamen existiert.

Bedenken Sie allerdings, dass die Massendaten üblicherweise *nicht* in der Nachricht vorhanden sind, wenn sich die Nachricht in einer Warteschlange befindet, und dass daher die Massendaten durch die Option MQGMO_CONVERT konvertiert werden.

Felder für MQRMH

Die MQRMH-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)

Gibt die Zeichensatzkennung der Massendaten an; dieses Attribut bezieht sich nicht auf Zeichendaten in der MQRMH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Folgende Sonderwerte sind zulässig:

MQCCSI_INHERIT
Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

Verwenden Sie nicht MQCCSI_INHERIT, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

Dieser Wert wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowsund WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

DataLogicalLength (MQLONG)

Das Feld *DataLogicalLength* gibt die Länge der Massendaten an, die von der MQRMH-Struktur referenziert werden.

Wenn die Massendaten tatsächlich in der Nachricht vorhanden sind, beginnen die Daten bei einem Offset von *StrucLength* Bytes ab dem Anfang der MQRMH-Struktur. Die Länge der gesamten Nachricht minus *StrucLength* gibt die Länge der vorhandenen Massendaten an.

Wenn Daten in der Nachricht vorhanden sind, gibt *DataLogicalLength* die Menge der relevanten Daten an. Normalerweise entspricht*DataLogicalLength* dem Wert der Länge der in der Nachricht vorhandenen Daten.

Wenn die MQRMH-Struktur die verbleibenden Daten im Objekt (ab dem angegebenen logischen Offset) darstellt, können Sie den Wert null für *DataLogicalLength* verwenden, sofern die Massendaten nicht tatsächlich in der Nachricht vorhanden sind.

Wenn keine Daten vorhanden sind, entspricht das Ende des MQRMH dem Ende der Nachricht.

Der Anfangswert dieses Feldes ist 0.

DataLogicalOffset (MQLONG)

Dieses Feld gibt den geringen Offset von Massendaten ab dem Start des Objektes an, zu dem die Massendaten gehören. Der Offset der Massendaten ab dem Objektanfang ist der sogenannte *logische Offset*. Dies ist *nicht* das physische Offset ab dem Anfang der MQRMH-Struktur; dieser Offset wird durch *StrucLength* angegeben.

Um das Senden großer Objekte mithilfe von Referenznachrichten zu ermöglichen, wird das logische Offset in zwei Felder aufgeteilt, und das tatsächliche logische Offset wird durch die Summe dieser zwei Felder angegeben:

- *DataLogicalOffset* stellt den Rest dar, den man erhält, wenn das logische Offset durch 1 000 000 000 dividiert wird. Es ist daher ein Wert im Bereich zwischen 0 und 999 999 999.
- *DataLogicalOffset2* stellt das Ergebnis dar, das man erhält, wenn das logische Offset durch 1 000 000 000 dividiert wird. Dieser Wert ist also die Anzahl vollständiger Vielfacher von 1 000 000 000, die im logischen Offset vorhanden sind. Die Anzahl Vielfache liegt im Bereich zwischen 0 und 999 999 999.

Der Anfangswert dieses Feldes ist 0.

DataLogicalOffset2 (MQLONG)

Dieses Feld gibt den hohen Offset der Massendaten ab dem Start des Objektes an, zu dem die Massendaten gehören. Dieser Wert liegt im Bereich von 0 bis 999 999 999. Weitere Informationen finden Sie im Abschnitt *DataLogicalOffset*.

Der Anfangswert dieses Feldes ist 0.

DestEnvLength (MQLONG)

Gibt die Länge der Zielumgebungsdaten an. Wenn dieses Feld null ist, gibt es keine Zielumgebungsdaten und *DestEnvOffset* wird ignoriert.

DestEnvOffset (MQLONG)

Dieses Feld gibt die relative Adresse der Zielumgebungsdaten ab dem Anfang der MQRMH-Struktur an. Zielumgebungsdaten können durch den Ersteller der Referenznachricht angegeben werden, sofern ihm diese Daten bekannt sind. Unter Windows beispielsweise können Zielumgebungsdaten der Verzeichnispfad des Objekts sein, in dem die Massendaten gespeichert werden sollen. Wenn der Ersteller allerdings die Zielumgebungsdaten nicht kennt, müssen die erforderlichen Umgebungsdaten vom benutzerdefinierten Nachrichtenexit bestimmt werden.

Die Länge der Zielumgebungsdaten wird durch *DestEnvLength* angegeben. Wenn diese Länge null ist, gibt es keine Zielumgebungsdaten und *DestEnvOffset* wird ignoriert. Falls vorhanden, müssen sich die Zielumgebungsdaten vollständig innerhalb der *StrucLength* Bytes am Anfang der Struktur befinden.

Anwendungen dürfen nicht davon ausgehen, dass die Zielumgebungsdaten mit den Daten, die von den Feldern *SrcEnvOffset*, *SrcNameOffset* und *DestNameOffset* adressiert werden, zusammenhängend sind.

Der Anfangswert dieses Feldes ist 0.

DestNameLength (MQLONG)

Dies ist die Länge des Zielobjektnamens. Wenn dieses Feld null ist, gibt es keinen Zielobjektnamen und *DestNameOffset* wird ignoriert.

DestNameOffset (MQLONG)

Dieses Feld gibt die relative Adresse des Zielobjektnamens ab dem Anfang der MQRMH-Struktur an. Der Zielobjektname kann durch den Ersteller der Referenznachricht angegeben werden, sofern ihm diese Daten bekannt sind. Wenn der Ersteller allerdings den Zielobjektnamen nicht kennt, muss das Objekt, das erstellt oder geändert werden soll, vom benutzerdefinierten Nachrichtenexit identifiziert werden.

Die Länge des Zielobjektnamens wird durch *DestNameLength* angegeben. Wenn diese Länge null ist, gibt es keinen Zielobjektnamen und *DestNameOffset* wird ignoriert. Falls vorhanden, muss sich der Zielobjektname vollständig innerhalb der *StrucLength* Bytes am Anfang der Struktur befinden.

Anwendungen dürfen nicht annehmen, dass der Zielobjektname mit Daten zusammenhängt, auf die sich die Felder *SrcEnvOffset*, *SrcNameOffset* und *DestEnvOffset* beziehen.

Der Anfangswert dieses Feldes ist 0.

Encoding (MQLONG)

Gibt die numerische Codierung der Massendaten an; dieses Attribut bezieht sich nicht auf numerische Daten in der MQRMH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Felds ist MQENC_NATIVE.

Flags (MQLONG)

Dies sind Referenznachrichtenflags. Die folgenden Flags sind definiert:

MQRMHF_LAST
Dieses Flag zeigt an, dass die Referenznachricht den letzten Teil des Referenzobjekts darstellt oder ihn enthält.

MQRMHF_NOT_LAST
Die Referenznachricht enthält nicht den letzten Teil des Objekts und stellt ihn nicht dar. MQRMHF_NOT_LAST wird zur Unterstützung der Programmdokumentation bereitgestellt. Diese Option ist nicht zur Verwendung mit einer anderen Option gedacht, da sie jedoch den Wert null hat, kann die Verwendung nicht erkannt werden.

Der Anfangswert dieses Felds ist MQRMHF_NOT_LAST.

Format (MQCHAR8)

Gibt den Formatnamen der Massendaten an.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds entsprechen denen für das Feld *Format* in MQMD.

Der Anfangswert dieses Felds ist MQFMT_NONE.

ObjectInstanceId (MQBYTE24)

Verwenden Sie dieses Feld, um eine bestimmte Instanz eines Objekts anzugeben. Wenn dies nicht benötigt wird, legen Sie es auf den folgenden Wert fest:

MQOII_NONE
Keine Objektinstanz-ID angegeben. Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQOII_NONE_ARRAY definiert. Sie hat den gleichen Wert wie MQOII_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_OBJECT_INSTANCE_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQOII_NONE.

ObjectType (MQCHAR8)

Dies ist ein Name, den der Nachrichtenexit verwenden kann, um Typen der Referenznachricht zu erkennen, die von ihm unterstützt werden. Der Name muss denselben Regeln wie das oben beschriebene Feld *Format* entsprechen.

Der Anfangswert dieses Feldes ist 8 Leerstellen.

SrcEnvLength (MQLONG)

Dies ist die Länge der Quellenumgebungsdaten. Wenn dieses Feld null ist, gibt es keine Quellenumgebungsdaten und *SrcEnvOffset* wird ignoriert.

Der Anfangswert dieses Feldes ist 0.

SrcEnvOffset (MQLONG)

Dieses Feld gibt die relative Adresse der Quellenumgebungsdaten ab dem Anfang der MQRMH-Struktur an. Quellenumgebungsdaten können durch den Ersteller der Referenznachricht angegeben werden, sofern ihm diese Daten bekannt sind. Unter Windows beispielsweise können Quellenumgebungsdaten der Verzeichnispfad des Objekts sein, das die Massendaten enthält. Wenn der Ersteller allerdings die Quellenumgebungsdaten nicht kennt, müssen erforderliche Umgebungsdaten vom benutzerdefinierten Nachrichtenexit bestimmt werden.

Die Länge der Quellenumgebungsdaten wird durch *SrcEnvLength* angegeben. Wenn diese Länge null ist, gibt es keine Quellenumgebungsdaten und *SrcEnvOffset* wird ignoriert. Falls vorhanden, müssen sich die Quellenumgebungsdaten vollständig innerhalb von *StrucLength* Bytes am Anfang der Struktur befinden.

Anwendungen dürfen nicht annehmen, dass die Umgebungsdaten unmittelbar nach dem letzten festen Feld in der Struktur anfangen oder dass sie mit Daten zusammenhängen, auf die sich die Felder *SrcNameOffset*, *DestEnvOffset* und *DestNameOffset* beziehen.

Der Anfangswert dieses Feldes ist 0.

SrcNameLength (MQLONG)

Dies ist die Länge des Quellenobjektnamens. Wenn dieses Feld null ist, gibt es keinen Quellenobjektnamen und *SrcNameOffset* wird ignoriert.

Der Anfangswert dieses Feldes ist 0.

SrcNameOffset (MQLONG)

Dieses Feld gibt die relative Adresse des Quellenobjektnamens ab dem Anfang der MQRMH-Struktur an. Der Quellenobjektname kann durch den Ersteller der Referenznachricht angegeben werden, sofern ihm diese Daten bekannt sind. Wenn der Ersteller allerdings den Quellenobjektnamen nicht kennt, muss das Objekt, auf das zugegriffen werden soll, vom benutzerdefinierten Nachrichtenexit identifiziert werden.

Die Länge des Quellenobjektnamens wird durch *SrcNameLength* angegeben. Wenn diese Länge null ist, gibt es keinen Quellenobjektnamen und *SrcNameOffset* wird ignoriert. Falls vorhanden, muss sich der Quellenobjektname vollständig innerhalb von *StrucLength* Bytes am Anfang der Struktur befinden.

Anwendungen dürfen nicht annehmen, dass der Quellenobjektname mit Daten zusammenhängt, auf die sich die Felder *SrcEnvOffset*, *DestEnvOffset* und *DestNameOffset* beziehen.

Der Anfangswert dieses Feldes ist 0.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQRMH_STRUC_ID
ID für die Struktur des Referenznachrichtheaders.

Für die Programmiersprache C ist auch die Konstante MQRMH_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQRMH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQRMH_STRUC_ID.

StrucLength (MQLONG)
Die Gesamtlänge von MQRMH, einschließlich der Zeichenfolgen am Ende der festen Felder, aber nicht der Massendaten.

Der Anfangswert dieses Felds ist null.

Version (MQLONG)

Die Versionsnummer der Struktur. Folgende Werte sind möglich:

MQRMH_VERSION_1
Struktur des Referenznachrichtenheaders der Version 1

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQRMH_CURRENT_VERSION
Aktuelle Version der Struktur Referenznachrichtenheader.

Der Anfangswert dieses Felds ist MQRMH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQRMH

Tabelle 542. Anfangswerte für Felder im MQRMH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQRMH_STRUC_ID	'RMH¬'
Version	MQRMH_VERSION_1	1
StrucLength	--	0
Encoding	MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQRMHF_NOT_LAST	0
ObjectType	--	Leerzeichen
ObjectInstanceId	MQOII_NONE	Nullen
SrcEnvLength	--	0
SrcEnvOffset	--	0
SrcNameLength	--	0
SrcNameOffset	--	0
DestEnvLength	--	0
DestEnvOffset	--	0
DestNameLength	--	0

Tabelle 542. Anfangswerte für Felder im MQRMH (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten
DestNameOffset	--	0
DataLogicalLength	--	0
DataLogicalOffset	--	0
DataLogicalOffset2	--	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQRMH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
  MQCHAR4   StrucId;             /* Structure identifier */
  MQLONG    Version;             /* Structure version number */
  MQLONG    StrucLength;         /* Total length of MQRMH, including
                                    strings at end of fixed fields, but
                                    not the bulk data */
  MQLONG    Encoding;            /* Numeric encoding of bulk data */
  MQLONG    CodedCharSetId;      /* Character set identifier of bulk
                                    data */
  MQCHAR8   Format;              /* Format name of bulk data */
  MQLONG    Flags;               /* Reference message flags */
  MQCHAR8   ObjectType;          /* Object type */
  MQBYTE24  ObjectInstanceId;    /* Object instance identifier */
  MQLONG    SrcEnvLength;        /* Length of source environment data */
  MQLONG    SrcEnvOffset;        /* Offset of source environment data */
  MQLONG    SrcNameLength;       /* Length of source object name */
  MQLONG    SrcNameOffset;       /* Offset of source object name */
  MQLONG    DestEnvLength;       /* Length of destination environment
                                    data */
  MQLONG    DestEnvOffset;       /* Offset of destination environment
                                    data */
  MQLONG    DestNameLength;      /* Length of destination object name */
  MQLONG    DestNameOffset;      /* Offset of destination object name */
  MQLONG    DataLogicalLength;   /* Length of bulk data */
  MQLONG    DataLogicalOffset;   /* Low offset of bulk data */
  MQLONG    DataLogicalOffset2;  /* High offset of bulk data */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQRMH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Total length of MQRMH,
                                         including strings at end of
                                         fixed fields, but not the bulk
                                         data */
  3 Encoding           fixed bin(31), /* Numeric encoding of bulk
                                         data */
  3 CodedCharSetId     fixed bin(31), /* Character set identifier of
                                         bulk data */
  3 Format             char(8),       /* Format name of bulk data */
  3 Flags              fixed bin(31), /* Reference message flags */
  3 ObjectType         char(8),       /* Object type */
  3 ObjectInstanceId   char(24),      /* Object instance identifier */
  3 SrcEnvLength       fixed bin(31), /* Length of source environment
                                         data */
  3 SrcEnvOffset       fixed bin(31), /* Offset of source environment
                                         data */
  3 SrcNameLength      fixed bin(31), /* Length of source object name */
  3 SrcNameOffset      fixed bin(31), /* Offset of source object name */
  3 DestEnvLength      fixed bin(31), /* Length of destination
                                         environment data */
  3 DestEnvOffset      fixed bin(31), /* Offset of destination
                                         environment data */
  3 DestNameLength     fixed bin(31), /* Length of destination object
                                         name */
  3 DestNameOffset     fixed bin(31), /* Offset of destination object
                                         name */
  3 DataLogicalLength  fixed bin(31), /* Length of bulk data */
  3 DataLogicalOffset  fixed bin(31), /* Low offset of bulk data */
  3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */
```

Deklaration in High Level Assembler

```
MQRMH                   DSECT
MQRMH_STRUCID           DS   CL4   Structure identifier
MQRMH_VERSION           DS   F     Structure version number
MQRMH_STRUCLENGTH       DS   F     Total length of MQRMH, including
*                                  strings at end of fixed fields, but
*                                  not the bulk data
MQRMH_ENCODING          DS   F     Numeric encoding of bulk data
MQRMH_CODEDCHARSETID    DS   F     Character set identifier of bulk
*                                  data
MQRMH_FORMAT            DS   CL8   Format name of bulk data
```

```
MQRMH_FLAGS               DS   F     Reference message flags
MQRMH_OBJECTTYPE          DS   CL8   Object type
MQRMH_OBJECTINSTANCEID    DS   XL24  Object instance identifier
MQRMH_SRCENVLENGTH        DS   F     Length of source environment data
MQRMH_SRCENVOFFSET        DS   F     Offset of source environment data
MQRMH_SRCNAMELENGTH       DS   F     Length of source object name
MQRMH_SRCNAMEOFFSET       DS   F     Offset of source object name
MQRMH_DESTENVLENGTH       DS   F     Length of destination environment
*                                     data
MQRMH_DESTENVOFFSET       DS   F     Offset of destination environment
*                                     data
MQRMH_DESTNAMELENGTH      DS   F     Length of destination object name
MQRMH_DESTNAMEOFFSET      DS   F     Offset of destination object name
MQRMH_DATALOGICALLENGTH   DS   F     Length of bulk data
MQRMH_DATALOGICALOFFSET   DS   F     Low offset of bulk data
MQRMH_DATALOGICALOFFSET2  DS   F     High offset of bulk data
*
MQRMH_LENGTH              EQU  *-MQRMH
                          ORG  MQRMH
MQRMH_AREA                DS   CL(MQRMH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQRMH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Total length of MQRMH, including'
                                 'strings at end of fixed fields, but'
                                 'not the bulk data'
  Encoding           As Long     'Numeric encoding of bulk data'
  CodedCharSetId     As Long     'Character set identifier of bulk data'
  Format             As String*8 'Format name of bulk data'
  Flags              As Long     'Reference message flags'
  ObjectType         As String*8 'Object type'
  ObjectInstanceId   As MQBYTE24 'Object instance identifier'
  SrcEnvLength       As Long     'Length of source environment data'
  SrcEnvOffset       As Long     'Offset of source environment data'
  SrcNameLength      As Long     'Length of source object name'
  SrcNameOffset      As Long     'Offset of source object name'
  DestEnvLength      As Long     'Length of destination environment'
                                 'data'
  DestEnvOffset      As Long     'Offset of destination environment'
                                 'data'
  DestNameLength     As Long     'Length of destination object name'
  DestNameOffset     As Long     'Offset of destination object name'
  DataLogicalLength  As Long     'Length of bulk data'
  DataLogicalOffset  As Long     'Low offset of bulk data'
  DataLogicalOffset2 As Long     'High offset of bulk data'
End Type
```

MQRR - Antwortdatensatz

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 543. Felder für MQRR

Feld	Beschreibung	Thema
CompCode	Beendigungscode für Warteschlange	CompCode
Reason	Ursachencode für Warteschlange	Reason

Überblick zu MQRR

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

Zweck: Verwenden Sie die MQRR-Struktur, um den Beendigungscode oder den Ursachencode zu empfangen, der aus der Open- oder Put-Operation für eine einzelne Zielwarteschlange resultiert, wenn das Ziel eine Verteilerliste ist. MQRR ist eine Ausgabestruktur für die Aufrufe MQOPEN, MQPUT und MQPUT1.

Zeichensatz und Codierung: Der Zeichensatz der Daten in MQRR muss dem entsprechen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, die Codierung der des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angebeben ist. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Verwendung: Indem ein Array dieser Strukturen bei den MQOPEN- und MQPUT-Aufrufen oder beim MQPUT1-Aufruf bereitgestellt wird, können Sie die Beendigungscodes und die Ursachencodes für alle Warteschlangen in einer Verteilerliste bestimmen, wenn das Ergebnis des Aufrufs uneinheitlich ist, das heißt, wenn der Aufruf bei einigen Warteschlangen erfolgreich ist, bei anderen nicht. Der Ursachencode MQRC_MULTIPLE_REASONS vom Aufruf gibt an, dass die Antwortdatensätze (falls von der Anwendung bereitgestellt) durch den Warteschlangenmanager festgelegt wurden.

Felder für MQRR

Die MQRR-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CompCode (MQLONG)

Dies ist der Beendigungscode, der aus der Open- oder Put-Operation für die Warteschlange mit dem Namen resultiert, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde.

Dies ist stets ein Ausgabefeld. Der Anfangswert dieses Felds ist MQCC_OK.

Ursache (MQLONG)

Dies ist der Ursachencode, der aus der Open- oder Put-Operation für die Warteschlange mit dem Namen resultiert, der durch das entsprechende, im MQOPEN- oder MQPUT1-Aufruf bereitgestellte Element im Array von MQOR-Strukturen angegeben wurde.

Dies ist stets ein Ausgabefeld. Der Anfangswert dieses Felds ist MQRC_NONE.

Anfangswerte und Sprachendeklarationen für MQRR

Tabelle 544. Anfangswerte für Felder im MQRR

Name des Felds	Name der Konstante	Wert der Konstanten
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0

Anmerkungen:

1. In der Programmiersprache C enthält die MakrovariableMQRR_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQRR MyRR = {MQRR_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
  MQLONG  CompCode;  /* Completion code for queue */
  MQLONG  Reason;    /* Reason code for queue */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */
```

Deklaration in Visual Basic

```
Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type
```

MQSCO - Optionen für die SSL-Konfiguration

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 545. Felder in MQSCO.

Liste der Felder in MQSCO, nach Version, mit Links zu den Abschnitten, in denen die Felder beschrieben werden.

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
KeyRepository	Speicherposition des Schlüsselrepository	KeyRepository
CryptoHardware	Einzeldaten der Verschlüsselungshardware	CryptoHardware
AuthInfoRecCount	Anzahl der vorhandenen MQAIR-Datensätze	AuthInfoRecCount
AuthInfoRecOffset	Offset des ersten MQAIR-Datensatzes ab dem Anfang des MQSCO	AuthInfoRecOffset
AuthInfoRecPtr	Adresse des ersten MQAIR-Datensatzes	AuthInfoRecPtr
Anmerkung: Die folgenden beiden Felder werden ignoriert, wenn *Version* kleiner als MQSCO_VERSION_2 ist.		
KeyResetCount	Anzahl der Rücksetzungen des geheimen SSL-Schlüssels	KeyResetCount
FipsRequired	Verwendung des FIPS-zertifizierten Verschlüsselungsalgorithmus in WebSphere MQ	„FipsRequired (MQLONG)" auf Seite 548
Anmerkung: Das folgende Feld wird ignoriert, wenn *Version* kleiner als MQSCO_VERSION_3 ist.		
EncryptionPolicySuiteB	Nur Suite B-Verschlüsselungsalgorithmen verwenden	EncryptionPolicySuiteB
Anmerkung: Das folgende Feld wird ignoriert, wenn *Version* kleiner als MQSCO_VERSION_4 ist.		
CertificateValPolicy	Prüfrichtlinie zertifizieren	CertificateValPolicy

Zugehörige Verweise

„MQCNO - Verbindungsoptionen” auf Seite 300
In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

„Überblick zu MQSCO” auf Seite 546
Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux und Windows -Clients.

„Felder für der MQSCO” auf Seite 546

„Anfangswerte und Sprachendeklarationen für MQSCO” auf Seite 550

Überblick zu MQSCO

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux und Windows -Clients.

Zweck: Die MQSCO-Struktur (zusammen mit den SSL-Feldern in der MQCD-Struktur) ermöglicht es einer Anwendung, die als WebSphere MQ MQI-Client ausgeführt wird, Konfigurationsoptionen anzugeben, die die Verwendung von SSL für die Client-Verbindung steuern, wenn das Kanalprotokoll TCP/IP ist. Die Struktur ist ein Eingabeparameter im MQCONNX-Anruf.

Wenn das Kanalprotokoll für den Clientkanal nicht TCP/IP ist, wird die MQSCO-Struktur ignoriert.

Zeichensatz und Codierung: Die Daten in MQSCO mustis sbe entsprechen dem Zeichensatz, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, und der Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angegeben wird.

Felder für der MQSCO

Die MQSCO-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

AuthInfoRecCount (MQLONG)

Dies ist die Anzahl der Authentifizierungsinformationsdatensätze (MQAIR), auf die das Feld *AuthInfoRecPtr* oder *AuthInfoRecOffset* verweisen. Weitere Informationen finden Sie im Abschnitt „MQAIR - Datensätze für Authentifizierungsinformationen” auf Seite 254. Der Wert muss null oder größer sein. Wenn der Wert nicht gültig ist, schlägt der Aufruf mit Ursachencode MQRC_AUTH_INFO_REC_COUNT_ERROR fehl.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

AuthInfoRecOffset (MQLONG)

Dies ist der Offset in Byte des ersten Authentifizierungsinformationsdatensatzes ab dem Anfang der MQSCO-Struktur. Der Offset kann positiv oder negativ sein. Das Feld wird ignoriert, wenn *AuthInfoRecCount* null ist.

Sie können entweder *AuthInfoRecOffset* oder *AuthInfoRecPtr* verwenden, um die MQAIR-Datensätze anzugeben, aber nicht beides. Informationen finden Sie in der Beschreibung des Felds *AuthInfoRecPtr*.

Dies ist ein Eingabefeld. Der Anfangswert dieses Feldes ist 0.

AuthInfoRecPtr (PMQAIR)

Dies ist die Adresse des ersten Datensatzes mit Authentifizierungsinformationen. Das Feld wird ignoriert, wenn *AuthInfoRecCount* null ist.

Sie können ein Array von MQAIR-Datensätzen auf zwei Arten bereitstellen:

- Durch Verwendung des Zeigerfelds *AuthInfoRecPtr*

  In diesem Fall kann die Anwendung ein von der MQSCO-Struktur separates Array von MQAIR-Datensätzen deklarieren und *AuthInfoRecPtr* auf die Adresse des Arrays festlegen.

Verwenden Sie *AuthInfoRecPtr* bei Programmiersprachen, die den Zeigerdatentyp so unterstützen, dass er in verschiedene Umgebungen portierbar ist (z. B. die Programmiersprache C).

- Durch Verwendung des Offsetfelds *AuthInfoRecOffset*

  In diesem Fall muss die Anwendung eine zusammengesetzte Struktur deklarieren, die eine MQSCO enthält, gefolgt von einem Array von MQAIR-Datensätzen, und *AuthInfoRecOffset* auf den Offset des ersten Datensatzes im Array ab dem Anfang der MQSCO-Struktur festlegen. Stellen Sie sicher, dass dieser Wert korrekt ist und dass es sich um einen Wert handelt, der in MQLONG aufgenommen werden kann (die restriktivste Programmiersprache ist COBOL, bei der der gültige Bereich von -999 999 999 bis +999 999 999 reicht).

  Verwenden Sie *AuthInfoRecOffset* bei Programmiersprachen, die den Zeigerdatentyp nicht unterstützen oder so implementieren, dass er nicht in verschiedene Umgebungen portierbar ist (z. B. die Programmiersprache COBOL).

Unabhängig vom ausgewählten Verfahren kann nur entweder *AuthInfoRecPtr* oder *AuthInfoRecOffset* verwendet werden. Der Aufruf schlägt mit Ursachencode MQRC_AUTH_INFO_REC_ERROR fehl, wenn beide ungleich null sind.

Dies ist ein Eingabefeld. Der Anfangswert dieses Felds ist in den Programmiersprachen, die Zeiger unterstützen, der Nullzeiger und in allen anderen Fällen eine vollständig auf null gesetzte Bytefolge.

Anmerkung: Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

CertificateValPolicy (MQLONG)

Dieses Feld gibt den Typ der verwendeten Zertifikatprüfrichtlinie an. Das Feld kann auf einen der folgenden Werte festgelegt werden:

MQ_CERT_VAL_POLICY_ANY
: Es werden alle Zertifikatprüfrichtlinien verwendet, die durch die Secure Sockets-Bibliothek unterstützt werden. Die Zertifikatskette wird akzeptiert, wenn eine der Richtlinien die Zertifikatskette als gültig bewertet.

MQ_CERT_VAL_POLICY_RFC5280
: Es wird nur die Zertifikatprüfrichtlinie verwendet, die dem Standard RFC 5280 entspricht. Bei dieser Einstellung erfolgt eine strengere Prüfung als bei der Einstellung "ANY", es werden aber einige ältere digitale Zertifikate zurückgewiesen.

Der Anfangswert dieses Felds ist MQ_CERT_VAL_POLICY_ANY.

CryptoHardware (MQCHAR256)
Dieses Feld gibt Konfigurationsdetails für die Verschlüsselungshardware an, die mit dem Clientsystem verbunden ist.

Sie können das Feld leer lassen, auf null setzen oder auf eine Zeichenfolge im folgenden Format festlegen:

```
GSK_PKCS11=<the PKCS #11 driver path and file name>;<the PKCS #11
token label>;<the PKCS #11 token password>;<symmetric cipher setting>;
```

Um Verschlüsselungshardware zu verwenden, die der PKCS #11-Schnittstelle entspricht (z. B. IBM 4960 oder IBM 4764), müssen die PKCS #11-Treiberpfad-, PKCS #11-Tokenbezeichnungs- und PKCS #11-Tokenkennwortzeichenfolgen angegeben werden, jeweils durch ein Semikolon abgeschlossen.

Der Treiberpfad für PKCS #11 bezeichnet einen absoluten Pfad zur gemeinsam genutzten Bibliothek, die die Unterstützung für die PKCS #11-Karte bereitstellt. Der Treiberdateiname für PKCS #11 bezeichnet den Namen der gemeinsam genutzten Bibliothek. Ein Beispiel für den erforderlichen Wert für den PKCS #11-Pfad und -Dateinamen ist:

```
/usr/lib/pkcs11/PKCS11_API.so
```

Die PKCS #11-Tokenbezeichnung darf nur Kleinbuchstaben enthalten. Wenn Sie Ihre Hardware mit einer in Tokenbezeichnung in Groß-/Kleinschreibung oder in Großschreibung konfiguriert haben, rekonfigurieren Sie sie mit dieser Bezeichnung in Kleinschreibung.

Wenn keine Konfiguration der Verschlüsselungshardware erforderlich ist, lassen Sie dieses Feld leer oder setzen es auf null.

Wenn der Wert kürzer als die Länge des Felds ist, beenden Sie den Wert mit einem Nullzeichen oder füllen Sie es mit Leerzeichen auf die Länge des Felds auf. Wenn der Wert nicht gültig ist oder zu einem Fehler führt, wenn er zum Konfigurieren der Verschlüsselungshardware verwendet wird, schlägt der Aufruf mit Ursachencode MQRC_CRYPTO_HARDWARE_ERROR fehl.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_SSL_CRYPTO_HARDWARE_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.

EncryptionPolicySuiteB (MQLONG)

Dieses Feld gibt an, ob eine Suite B-kompatible Verschlüsselung verwendet wird und welche Stufe angewandt wird. Der Wert kann einem oder mehreren der folgenden Werte entsprechen:

- MQ_SUITE_B_NONE

  Suite B-kompatible Verschlüsselung wird nicht verwendet.
- MQ_SUITE_B_128_BIT

  Sicherheit für Suite B 128-Bit-Stufe wird verwendet.
- MQ_SUITE_B_192_BIT

  Sicherheit für Suite B 192-Bit-Stufe wird verwendet

Anmerkung: Das Verwenden von MQ_SUITE_B_NONE mit einem anderen Wert in diesem Feld ist ungültig.

FipsRequired (MQLONG)
WebSphere MQ kann mit Verschlüsselungshardware konfiguriert werden, sodass die vom Hardwareprodukt bereitgestellten Verschlüsselungsmodule verwendet werden. Dabei kann es sich um (bis zu einem bestimmten FIPS-Level) FIPS-zertifizierte Module handeln, abhängig von der verwendeten Verschlüsselungshardware. Verwenden Sie dieses Feld, um anzugeben, dass nur FIPS-zertifizierte Algorithmen verwendet werden, wenn die Verschlüsselung in der von WebSphere MQbereitgestellten Software bereitgestellt wird.

Bei der Installation von WebSphere MQ wird auch eine Implementierung der SSL-Verschlüsselung installiert, die einige FIPS-zertifizierte Module bereitstellt.

Folgende Werte stehen zur Auswahl:

MQSSL_FIPS_NO
Dies ist der Standardwert. Die Angabe dieses Werts bewirkt Folgendes:

- Jede auf einer Plattform unterstützte CipherSpec kann verwendet werden.
- Bei Ausführung ohne Verschlüsselungshardware werden unter Verwendung der FIPS 140-2-zertifizierten Verschlüsselung auf den WebSphere MQ-Plattformen die folgenden CipherSpecs ausgeführt:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES
Die Angabe dieses Werts bewirkt Folgendes (sofern keine Verschlüsselungshardware für die Verschlüsselung verwendet wird):

- In der CipherSpec für diese Clientverbindung können nur FIPS-zertifizierte Verschlüsselungsalgorithmen verwendet werden.

- Ein- und abgehende SSL-Kanalverbindungen sind nur bei Verwendung einer der folgenden CipherSpecs erfolgreich:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

Anmerkungen:

1. Die CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA wird nicht weiter unterstützt.
2. Wenn die Verwendung von FIPS-zertifizierten CipherSpecs konfiguriert ist, lehnt der MQI-Client nach Möglichkeit Verbindungen, in denen eine nicht FIPS-zertifizierte CipherSpec angegeben ist, mit dem Ursachencode MQRC_SSL_INITIALIZATION_ERROR ab. Es kann nicht garantiert werden, dass WebSphere MQ alle Verbindungen dieser Art ablehnt. Es liegt in der eigenen Verantwortung des Kunden, zu ermitteln, ob die WebSphere MQ-Konfiguration mit FIPS kompatibel ist.

distributed *KeyRepository (MQCHAR256)*

Dieses Feld ist nur für WebSphere MQ MQI-Clients relevant, die auf UNIX-, Linux-und Windows -Systemen ausgeführt werden. Es gibt die Position der Schlüsseldatenbankdatei an, in der Schlüssel und Zertifikate gespeichert werden. Der Name der Schlüsseldatenbankdatei muss das Format `zzz.kdb` haben, wobei `zzz` benutzerdefiniert ist. Das Feld *KeyRepository* enthält den Pfad zu dieser Datei zusammen mit dem Dateinamensstamm (alle Zeichen im Dateinamen bis einschließlich `.kdb`). Das Dateisuffix `.kdb` wird automatisch hinzugefügt.

Jede Schlüsseldatenbankdatei hat eine zugeordnete *Kennwortstashdatei*. Diese enthält codierte Kennwörter, die programmgesteuerten Zugriff auf die Schlüsseldatenbank ermöglichen. Die Kennwortstashdatei muss sich im selben Verzeichnis wie die Schlüsseldatenbank befinden, denselben Dateistamm haben und mit dem Suffix `.sth` enden.

Wenn das Feld *KeyRepository* beispielsweise den Wert `/xxx/yyy/key` hat, muss die Schlüsseldatenbankdatei `/xxx/yyy/key.kdb` sein und die Kennwortstashdatei muss `/xxx/yyy/key.sth` sein, wobei `xxx` und `yyy` Verzeichnisnamen darstellen.

Wenn der Wert kürzer als die Länge des Felds ist, beenden Sie den Wert mit einem Nullzeichen oder füllen Sie es mit Leerzeichen auf die Länge des Felds auf. Der Wert wird nicht geprüft. Wenn es einen Fehler beim Zugriff auf das Schlüsselrepository gibt, schlägt der Aufruf mit Ursachencode MQRC_KEY_REPOSITORY_ERROR fehl.

Um eine SSL-Verbindung von einem WebSphere MQ MQI-Client aus auszuführen, setzen Sie *KeyRepository* auf einen gültigen Namen einer Schlüsseldatenbankdatei.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_SSL_KEY_REPOSITORY_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.

KeyResetCount (MQLONG)
Dies stellt die Gesamtzahl unverschlüsselter Bytes dar, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL- oder TLS-Dialog gesendet bzw. empfangen werden.

Die Anzahl der Byte enthält Steuerinformationen, die vom MCA gesendet wurden.

Wenn Sie für die Anzahl der Rücksetzungen von geheimen SSL- oder TLS-Schlüsseln einen Wert im Bereich von 1 Byte bis 32 KB setzen, verwenden die SSL- bzw. TLS-Kanäle als Zählerstand für die Rücksetzung des geheimen Schlüssels 32 KB. Dadurch werden die Verarbeitungskosten für übermäßig viele Schlüsselrücksetzungen vermieden, wie es bei kleinen Rücksetzungswerten für geheime SSL- oder TSL-Schlüssel der Fall wäre.

Dies ist ein Eingabefeld. Der Wert ist eine Zahl im Bereich von 0 bis 999 999 999, wobei der Standardwert 0 ist. Verwenden Sie den Wert 0, um anzugeben, dass geheime Schlüssel nie neu vereinbart werden.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQSCO_STRUC_ID
ID für die Struktur der SSL-Konfigurationsoptionen.

Für die Programmiersprache C ist auch die Konstante MQSCO_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQSCO_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSCO_STRUC_ID.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQSCO_VERSION_1
Struktur der SSL-Konfigurationsoptionen der Version 1.

MQSCO_VERSION_2
Struktur der SSL-Konfigurationsoptionen der Version 2.

MQSCO_VERSION_3
Struktur der SSL-Konfigurationsoptionen der Version 3.

MQSCO_VERSION_4
Struktur der SSL-Konfigurationsoptionen der Version 4.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQSCO_CURRENT_VERSION
Aktuelle Version der Struktur der SSL-Konfigurationsoptionen.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSCO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQSCO

Tabelle 546. Anfangswerte der Felder in MQSCO.

Beschreibung der Felder in MQSCO und ihrer Anfangswerte

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQSCO_STRUC_ID	'SCO¬'
Version	MQSCO_CURRENT_VERSION	1
KeyRepository	--	Nullzeichenfolge oder Leerzeichen.
CryptoHardware	--	Nullzeichenfolge oder Leerzeichen.
AuthInfoRecCount	--	0
AuthInfoRecOffset	--	0
AuthInfoRecPtr	--	Nullzeiger oder Null Byte
KeyResetCount	MQSCO_RESET_COUNT_DEFAULT	0
FipsRequired	MQSSL_FIPS_NO	0
EncryptionPolicy-SuiteB	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0

Tabelle 546. Anfangswerte der Felder in MQSCO. Beschreibung der Felder in MQSCO und ihrer Anfangswerte *(Forts.)*		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**
CertificateValPolicy	MQ_CERT_VAL_POLICY_DEFAULT	0

Notes:

1. The symbol ¬ represents a single blank character.
2. In the C programming language, the macro variable MQSCO_DEFAULT contains the values listed above. Use it in the following way to provide initial values for the fields in the structure:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
   MQCHAR4    StrucId;                    /* Structure identifier */
   MQLONG     Version;                    /* Structure version number */
   MQCHAR256  KeyRepository;              /* Location of SSL key */
                                          /* repository */
   MQCHAR256  CryptoHardware;             /* Cryptographic hardware */
                                          /* configuration string */
   MQLONG     AuthInfoRecCount;           /* Number of MQAIR records */
                                          /* present */
   MQLONG     AuthInfoRecOffset;          /* Offset of first MQAIR */
                                          /* record from start of */
                                          /* MQSCO structure */
   PMQAIR     AuthInfoRecPtr;             /* Address of first MQAIR */
                                          /* record */
/* Ver:1 */
   MQLONG     KeyResetCount;              /* Number of unencrypted */
                                          /* bytes sent/received */
                                          /* before secret key is */
                                          /* reset */
   MQLONG     FipsRequired;               /* Using FIPS-certified */
/* Ver:2 */
                                          /* algorithms */
   MQLONG     EncryptionPolicySuiteB[4];  /* Use only Suite B */
/* Ver:3 */
                                          /* cryptographic algorithms */
   MQLONG     CertificateValPolicy;       /* Certificate validation */
                                          /* policy */
/* Ver:4 */
```

COBOL-Delaration

```
**   MQSCO structure
  10 MQSCO.
**    Structure identifier
   15 MQSCO-STRUCID                 PIC X(4).
**    Structure version number
   15 MQSCO-VERSION                 PIC S9(9) BINARY.
**    Location of SSL key repository
   15 MQSCO-KEYREPOSITORY           PIC X(256).
**    Cryptographic hardware configuration string
   15 MQSCO-CRYPTOHARDWARE          PIC X(256).
**    Number of MQAIR records present
   15 MQSCO-AUTHINFORECCOUNT        PIC S9(9) BINARY.
**    Offset of first MQAIR record from start of MQSCO structure
   15 MQSCO-AUTHINFORECOFFSET       PIC S9(9) BINARY.
** Address of first MQAIR record
    15  MQSCO-AUTHINFORECPTR POINTER.
** Version 1 **
** Number of unencrypted bytes sent/received before secret key is
** reset
```

```
15  MQSCO-KEYRESETCOUNT PIC S9(9) BINARY.
** Using FIPS-certified algorithms
15  MQSCO-FIPSREQUIRED PIC S9(9) BINARY.
** Version 2 **
** Use only Suite B cryptographic algorithms
15  MQSCO-ENCRYPTIONPOLICYSUITEB PIC S9(9) BINARY OCCURS 4.
** Version 3 **
** Certificate validation policy setting
15  MQSCO-CERTIFICATEVALPOLICY PIC S9(9) BINARY.
** Version 4
```

Deklaration in PL/I

```
dcl
 1 MQSCO based,
  3 StrucId                   char(4),       /* Structure identifier */
  3 Version                   fixed bin(31), /* Structure version number */
  3 KeyRepository             char(256),     /* Location of SSL key
                                                repository */
  3 CryptoHardware            char(256),     /* Cryptographic hardware
                                                configuration string */
  3 AuthInfoRecCount          fixed bin(31), /* Number of MQAIR records
                                                present */
  3 AuthInfoRecOffset         fixed bin(31), /* Offset of first MQAIR record
                                                from start of MQSCO structure */
  3 AuthInfoRecPtr            pointer,       /* Address of first MQAIR record */
  3 KeyResetCount             fixed bin(31), /* Key reset count */
/* Version 1 */
  3 FipsRequired              fixed bin(31), /* FIPS required */
/* Version 2 */
  3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
  3 CertificateValPolicy      fixed bin(31); /* Certificate validation policy */
/* Version 4 */
```

Deklaration in Visual Basic

```
Type MQSCO
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  KeyRepository     As String*256 'Location of SSL key repository'
  CryptoHardware    As String*256 'Cryptographic hardware configuration'
                                  'string'
  AuthInfoRecCount  As Long       'Number of MQAIR records present'
  AuthInfoRecOffset As Long       'Offset of first MQAIR record from'
                                  'start of MQSCO structure'
  AuthInfoRecPtr    As MQPTR      'Address of first MQAIR record'
  KeyResetCount     As Long       'Number of unencrypted bytes sent/received before secret key
is reset'
'Version 1'
  FipsRequired      As Long       'Mandatory FIPS CipherSpecs?'
'Version 2'
End Type
```

MQSD - Subskriptionsdeskriptor

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur.

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options
ObjectName	Objektname	ObjectName
AlternateUserId	Alternative Benutzer-ID	AlternateUserId
AlternateSecurityId	Alternative Sicherheits-ID	AlternateSecurityId
SubExpiry	Ablaufzeit der Subskription	SubExpiry

Feld	Beschreibung	Thema
ObjectString	Objektzeichenfolge	ObjectString
SubName	Subskriptionsname	SubName
SubUserData	Subskriptionsbenutzerdaten	SubUserData
SubCorrelId	Korrelations-ID der Subskription	SubCorrelId
PubPriority	Veröffentlichungspriorität	PubPriority
PubAccountingToken	Veröffentlichungsabrechnungstoken	PubAccountingToken
PubAppIdentityData	Identitätsdaten der Veröffentlichungsanwendung	PubAppIdentityData
SelectionString	Zeichenfolge mit Auswahlkriterien	SelectionString
SubLevel	Subskriptionsebene	SubLevel
ResObjectString	Langer Objektname	ResObjectString

Überblick für MQSD

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSsowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQSD-Struktur wird verwendet, um Details zur vorgenommenen Subskription anzugeben.

Die Struktur ist ein Ein-/Ausgabeparameter im Aufruf MQSUB. Weitere Informationen finden Sie unter Hinweise zur Verwendung von MQSUB.

Verwaltete Subskriptionen: Wenn eine Anwendung nicht eine bestimmte Warteschlange als Ziel für die Veröffentlichungen verwenden muss, die mit ihrer Subskription übereinstimmen, kann sie die Funktion für verwaltete Subskriptionen verwenden. Wird von einer Anwendung eine verwaltete Subskription verwendet, informiert der Warteschlangenmanager den Abonnenten über das Ziel, zu dem die veröffentlichten Nachrichten gesendet werden, indem er eine Objektkennung als Ausgabe des MQSUB-Aufrufs bereitstellt. Weitere Informationen hierzu finden Sie unter Hobj (MQHOBJ) - Eingabe/Ausgabe.

Wird die Subskription entfernt, beseitigt der Warteschlangenmanager in den nachstehenden Situationen auch die Nachrichten, die vom verwalteten Ziel nicht abgerufen wurden:

- Wenn die Subskription entfernt wird - durch Verwendung von MQCLOSE mit MQCO_REMOVE_SUB - und die verwaltete Kennung "Hobj" geschlossen wird.
- Durch implizite Verfahren, wenn die Verbindung zu einer Anwendung verloren geht, unter Verwendung einer nicht permanenten Subskription (MQSO_NON_DURABLE)
- Durch Ablauf, wenn eine Subskription entfernt wird, weil sie abgelaufen ist, und der verwaltete Hobj geschlossen wird

Sie müssen verwaltete Subskriptionen mit nicht permanenten Subskriptionen verwenden, damit diese Bereinigung erfolgen kann und damit Nachrichten für geschlossene nicht permanente Subskriptionen keinen Speicherplatz in Ihrem Warteschlangenmanager beanspruchen. Dauerhafte Subskriptionen können auch verwaltete Ziele verwenden.

Version: Die aktuelle Version von MQSD ist MQSD_VERSION_1.

Zeichensatz und Codierung: Daten in MQSD müssen im Zeichensatz vorliegen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird, sowie in der Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQSD

Die MQSD-Struktur enthält die folgenden Felder, die in alphabetischer Reihenfolge beschrieben werden:

AlternateSecurityId (MQBYTE40)

Dies ist eine Sicherheits-ID, die mit der AlternateUserId an den Berechtigungsservice übergeben wird, damit entsprechende Berechtigungsprüfungen ausgeführt werden können.

AlternateSecurityId wird nur verwendet, wenn MQSO_ALTERNATE_USER_AUTHORITY angegeben ist und das Feld AlternateUserId nicht bis zum ersten Nullzeichen oder bis zum Ende des Felds vollständig leer ist.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME bleibt dieses Feld unverändert.

Weitere Informationen finden Sie in der Beschreibung zu „AlternateSecurityId (MQBYTE40)" auf Seite 464 im MQOD-Datentyp.

AlternateUserId (MQCHAR12)

Wenn Sie MQSO_ALTERNATE_USER_AUTHORITY angeben, enthält dieses Feld eine alternative Benutzer-ID, die anstelle der Benutzer-ID, mit der die Anwendung derzeit ausgeführt wird, für die Subskription und für die Ausgabe an die Zielwarteschlange (die im Parameter *Hobj* des MQSUB-Aufrufs angegeben ist) verwendet wird.

Ist dies erfolgreich, wird die in diesem Feld angegebene Benutzer-ID anstelle der Benutzer-ID, mit der die Anwendung derzeit ausgeführt wird, als die Benutzer-ID aufgezeichnet, die Eigner der Subskription ist.

Wenn MQSO_ALTERNATE_USER_AUTHORITY angegeben wird und dieses Feld bis zum ersten Nullzeichen oder bis zum Ende des Felds vollständig leer ist, kann die Subskription nur erfolgreich ausgeführt werden, wenn zum Subskribieren dieses Themas mit den angegebenen Optionen oder der Zielwarteschlange für die Ausgabe keine Benutzerberechtigung erforderlich ist.

Wenn MQSO_ALTERNATE_USER_AUTHORITY nicht angegeben wird, wird dieses Feld ignoriert.

Für die angegebenen Umgebungen gelten die folgenden Unterschiede:

- Unter z/OS werden nur die ersten 8 Zeichen von AlternateUserId verwendet, um die Berechtigung für die Subskription zu prüfen. Die aktuelle Benutzer-ID muss jedoch zur Angabe dieser bestimmten alternativen Benutzer-ID berechtigt sein. Alle 12 Zeichen der alternativen Benutzer-ID werden für diese Prüfung verwendet. Die Benutzer-ID darf nur vom externen Sicherheitsmanager erlaubte Zeichen enthalten.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME bleibt dieses Feld unverändert.

Dies ist ein Eingabefeld. Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 12 leere Zeichen in anderen Programmiersprachen.

ObjectName (MQCHAR48)

Dies ist der Name des Themenobjekts, wie es im lokalen Warteschlangenmanager definiert ist.

Der Name kann folgende Zeichen enthalten:

- Großbuchstaben (A bis Z)
- Kleinbuchstaben des Alphabets(a bis z)
- Numerische Ziffern (0 bis 9)
- Punkt (.), Schrägstrich (/), Unterstrich (_), Prozent (%)

Der Name darf keine führenden oder eingebetteten Leerzeichen enthalten, wohl aber abschließende Leerzeichen. Verwenden Sie ein Nullzeichen zur Kennzeichnung des Endes signifikanter Daten im Namen;

die Null und alle ihr folgenden Zeichen werden wie Leerzeichen behandelt. Die folgenden Beschränkungen gelten für die angegebenen Umgebungen:

- Bei Systemen, die EBCDIC Katakana verwenden, können keine Kleinbuchstaben verwendet werden.
- Unter z/OS:
  - Vermeiden Sie Namen, die mit einem Unterstrich beginnen oder enden; sie können von den Betriebs- und Steuerkonsolen nicht verarbeitet werden.
  - Das Prozentzeichen hat eine spezielle Bedeutung für RACF. Wenn RACF als externer Sicherheitsmanager verwendet wird, dürfen Namen kein Prozentzeichen enthalten. Andernfalls werden diese Namen nicht in Sicherheitsprüfungen einbezogen, wenn generische RACF-Profile verwendet werden.
- In IBM i müssen innerhalb von Befehlen vorkommende Namen, die Kleinbuchstaben, Schrägstriche oder Prozentzeichen enthalten, zwischen An- und Abführungszeichen stehen. Diese Anführungszeichen dürfen nicht bei Namen angegeben werden, die Felder in Strukturen oder Parameter bei Aufrufen sind.

Der *ObjectName* wird verwendet, um den vollständigen Themennamen zu bilden.

Der vollständige Abschnittsname kann aus zwei verschiedenen Feldern erstellt werden: *ObjectName* und *ObjectString*. Ausführliche Informationen über die Verwendung dieser beiden Felder finden Sie im Abschnitt „Verwenden von Themenzeichenfolgen” auf Seite 569.

Wenn das durch das Feld *ObjectName* angegebene Objekt nicht gefunden wird, schlägt der Aufruf mit dem Ursachencode MQRC_UNKNOWN_OBJECT_NAME fehl, auch wenn eine in *ObjectString* angegebene Zeichenfolge vorhanden ist.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung der Option MQSO_RESUME bleibt dieses Feld unverändert.

Die Länge dieses Felds wird durch MQ_TOPIC_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann der Name des subskribierten Themenobjekts nicht geändert werden. Dieses Feld und das Feld *ObjectString* können übergangen werden. Wenn sie bereitgestellt werden, müssen sie in denselben vollständigen Themennamen aufgelöst werden. Andernfalls schlägt der Aufruf mit MQRC_TOPIC_NOT_ALTERABLE fehl.

ObjectString (MQCHARV)

Dies ist der zu verwendende ausgeschriebene Objektname.

Die *ObjectString* wird verwendet, um den vollständigen Themennamen zu bilden.

Der vollständige Abschnittsname kann aus zwei verschiedenen Feldern erstellt werden: *ObjectName* und *ObjectString*. Ausführliche Informationen über die Verwendung dieser beiden Felder finden Sie im Abschnitt „Verwenden von Themenzeichenfolgen” auf Seite 569.

Die maximale Länge von *ObjectString* beträgt 10240.

Wenn *ObjectString* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der MQCHARV-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_OBJECT_STRING_ERROR fehl.

Dies ist ein Eingabefeld. Die Anfangswerte der Felder in dieser Struktur entsprechen denen der MQCHARV-Struktur.

Wenn Platzhalter in der *ObjectString* enthalten sind, kann die Interpretation dieser Platzhalter über die Platzhalteroptionen gesteuert werden, die im Options-Feld des MQSD angegeben sind.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung der Option MQSO_RESUME bleibt dieses Feld unverändert. Der vollständige Themenname wird im Feld *ResObjectString* zurückgegeben, wenn ein Puffer bereitgestellt wird.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann der ausgeschriebene Name des subskribierten Themenobjekts nicht geändert werden. Dieses Feld und das Feld *ObjectNa-*

me können übergangen werden. Wenn sie bereitgestellt werden, müssen sie in denselben vollständigen Themennamen aufgelöst werden oder der Aufruf schlägt mit MQRC_TOPIC_NOT_ALTERABLE fehl.

Optionen (MQLONG)
Stellt Optionen zur Steuerung der Aktion des Aufrufs MQSUB bereit.

Sie müssen mindestens eine der folgenden Optionen angeben:

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

Die Werte, die Sie für die Optionen angeben, können wie folgt verwendet werden:

- Die Werte können hinzugefügt werden. Fügen Sie dieselbe Konstante nur einmal hinzu.
- Die Werte können mithilfe der bitweisen ODER-Operation kombiniert werden, falls die Programmiersprache bitweise Operationen unterstützt.

Auf ungültige Kombinationen wird in diesem Thema hingewiesen: alle anderen Kombinationen sind gültig.

Zugriffs- oder Erstellungsoptionen: Zugriffs- und Erstellungsoptionen legen fest, ob eine Subskription erstellt wird oder ob eine vorhandene Subskription zurückgegeben oder geändert wird. Sie müssen mindestens eine dieser Optionen angeben. In der Tabelle sind gültige Kombinationen für Zugriffs- und Erstellungsoptionen aufgeführt.

Kombination von Optionen	**Anmerkungen**
MQSO_CREATE	Erstellt eine Subskription, wenn keine vorhanden ist. Diese Kombination schlägt fehl, wenn die Subskription vorhanden ist.
MQSO_RESUME	Setzt eine vorhandene Subskription fort. Diese Kombination schlägt fehl, wenn keine Subskription vorhanden ist.
MQSO_CREATE + MQSO_RESUME	Erstellt eine Subskription, falls keine existiert, und nimmt eine übereinstimmende wieder auf, falls sie existiert. Diese Kombination ist nützlich, wenn sie in einer Anwendung verwendet wird, die mehrmals ausgeführt wird.
MQSO_ALTER (siehe Hinweis)	Setzt eine vorhandene Subskription fort, wobei alle Felder so geändert werden, dass sie mit den im MQSD angegebenen Werten übereinstimmen. Diese Kombination schlägt fehl, wenn keine Subskription vorhanden ist.
MQSO_CREATE + MQSO_ALTER (siehe Hinweis)	Erstellt eine Subskription, wenn keine vorhanden ist, und setzt eine übereinstimmende Subskription fort, wenn eine vorhanden ist, wobei alle Felder so geändert werden, dass sie mit den im MQSD angegebenen Werten übereinstimmen. Diese Kombination ist nützlich, wenn sie in einer Anwendung verwendet wird, die sicherstellen möchte, dass sich ihre Subskription in einem bestimmten Status befindet, bevor die Ausführung fortgesetzt wird.

Kombination von Optionen	Anmerkungen
Hinweis: Optionen, die MQSO_ALTER angeben, können auch MQSO_RESUME angeben, diese Kombination hat jedoch keine zusätzliche Auswirkung zur alleinigen Angabe von MQSO_ALTER. MQSO_ALTER schließt MQSO_RESUME ein, da der Aufruf von MQSUB zum Ändern einer Subskription impliziert, dass die Subskription auch fortgesetzt wird. Das Gegenteil trifft jedoch nicht zu: Die Wiederaufnahme einer Subskription impliziert nicht, dass sie geändert werden muss.	

MQSO_CREATE

Erstellt eine neue Subskription für das angegebene Thema. Wenn eine Subskription vorhanden ist, die denselben *SubName* verwendet, schlägt der Aufruf mit MQRC_SUB_ALREADY_EXISTS fehl. Dieser Fehler kann vermieden werden, indem die Option MQSO_CREATE mit MQSO_RESUME kombiniert wird. Der *SubName* ist nicht immer erforderlich. Weitere Informationen finden Sie in der Beschreibung zu diesem Feld.

Die Kombination von MQSO_CREATE mit MQSO_RESUME gibt eine Kennung für eine bereits vorhandene Subskription für den angegebenen *SubName* zurück, sofern eine gefunden wird. Wenn keine Subskription vorhanden ist, wird unter Verwendung aller im MQSD bereitgestellten Felder eine neue erstellt.

MQSO_CREATE kann auch mit MQSO_ALTER kombiniert werden, was eine ähnliche Auswirkung hat.

MQSO_RESUME

Gibt eine Kennung einer bereits vorhandenen Subskription zurück, die mit der durch *SubName* angegebenen Subskription übereinstimmt. An den Attributen der übereinstimmenden Subskriptionen werden keine Änderungen vorgenommen und sie werden bei der Ausgabe in der MQSD-Struktur zurückgegeben. Nur die folgenden MQSD-Felder werden verwendet: StrucId, Version, Options, AlternateUserId und AlternateSecurityId und SubName.

Der Aufruf schlägt mit dem Ursachencode MQRC_NO_SUBSCRIPTION fehl, wenn keine Subskription vorhanden ist, die mit dem vollständigen Subskriptionsnamen übereinstimmt. Dieser Fehler kann vermieden werden, indem die Option MQSO_CREATE mit MQSO_RESUME kombiniert wird.

Die Benutzer-ID der Subskription ist die Benutzer-ID, von der sie erstellt wurde, oder, wenn sie später von einer anderen Benutzer-ID geändert wurde, die Benutzer-ID der letzten erfolgreichen Änderung. Wenn eine AlternateUserId verwendet wird und die Verwendung alternativer Benutzer-IDs für diesen Benutzer zulässig ist, wird die alternative Benutzer-ID als die Benutzer-ID aufgezeichnet, die die Subskription erstellt hat, und nicht die Benutzer-ID, unter der die Subskription erstellt wurde.

Wenn eine übereinstimmende Subskription vorhanden ist, die ohne die Option MQSO_ANY_USERID erstellt wurde, und die Benutzer-ID der Subskription von der der Anwendung abweicht, die eine Kennung für die Subskription anfordert, schlägt der Aufruf mit dem Ursachencode MQRC_IDENTITY_MISMATCH fehl.

Wenn eine übereinstimmende Subskription vorhanden ist und derzeit verwendet wird, schlägt der Aufruf mit MQRC_SUBSCRIPTION_IN_USE fehl.

Wenn die in SubName benannte Subskription keine gültige Subskription für die Fortsetzung oder Änderung aus einer Anwendung ist, schlägt der Aufruf mit MQRC_INVALID_SUBSCRIPTION fehl.

MQSO_RESUME wird durch MQSO_ALTER impliziert, sodass eine Kombination mit dieser Option nicht erforderlich ist. Eine Kombination der zwei Optionen verursacht jedoch keinen Fehler.

MQSO_ALTER

Gibt eine Kennung einer bereits vorhandenen Subskription zurück, deren vollständiger Subskriptionsname mit dem in *SubName* angegebenen Namen übereinstimmt. Alle Attribute der Subskription, die von den im MQSD angegebenen Werten abweichen, werden in der Subskription geändert, es sei denn, eine Änderung ist für dieses Attribut nicht zugelassen. Details finden Sie in der Beschreibung der einzelnen Attribute sowie in der nachstehenden Tabelle. Wenn Sie versuchen, eine Attribut zu ändern,

das nicht geändert werden kann, oder eine Subskription zu ändern, für die die Option MQSO_IMMUTABLE festgelegt ist, schlägt der Aufruf mit dem in der folgenden Tabelle aufgeführten Ursachencode fehl.

Der Aufruf schlägt mit dem Ursachencode MQRC_NO_SUBSCRIPTION fehl, wenn keine Subskription vorhanden ist, die mit dem vollständigen Subskriptionsnamen übereinstimmt. Sie können diesen Fehler vermeiden, indem Sie die Option MQSO_CREATE mit MQSO_ALTER kombinieren.

Die Kombination von MQSO_CREATE mit MQSO_ALTER gibt eine Kennung für eine bereits vorhandene Subskription für den angegebenen *SubName* zurück, sofern eine gefunden wird. Wenn keine Subskription vorhanden ist, wird unter Verwendung aller im MQSD bereitgestellten Felder eine neue erstellt.

Die Benutzer-ID der Subskription ist die Benutzer-ID, die die Subskription erstellt hat. Wurde die Subskription später von einer anderen Benutzer-ID geändert, ist dies die die Benutzer-ID der letzten erfolgreichen Änderung. Wenn eine AlternateUserId verwendet wird und die Verwendung alternativer Benutzer-IDs für diesen Benutzer zulässig ist, wird die alternative Benutzer-ID als die Benutzer-ID aufgezeichnet, die die Subskription erstellt hat, und nicht die Benutzer-ID, unter der die Subskription erstellt wurde.

Wenn eine übereinstimmende Subskription vorhanden ist, die ohne die Option MQSO_ANY_USERID erstellt wurde, und die Benutzer-ID der Subskription von der der Anwendung abweicht, die eine Kennung für die Subskription anfordert, schlägt der Aufruf mit dem Ursachencode MQRC_IDENTITY_MISMATCH fehl.

Wenn eine übereinstimmende Subskription vorhanden ist und derzeit verwendet wird, schlägt der Aufruf mit MQRC_SUBSCRIPTION_IN_USE fehl.

Wenn die in SubName benannte Subskription keine gültige Subskription für die Fortsetzung oder Änderung aus einer Anwendung ist, schlägt der Aufruf mit MQRC_INVALID_SUBSCRIPTION fehl.

In der folgenden Tabelle ist die Fähigkeit von MQSO_ALTER dargestellt, Attributwerte in MQSD und MQSUB zu ändern.

Tabelle 547. Attribute in MQSD und MQSUB, die geändert werden können

Datentypdeskriptor oder Funktionsaufruf	Name des Felds	Kann dieses Attribut mit MQSO_ALTER geändert werden	Ursachencode
MQSD	Dauerhaftigkeitsoption	Nein	MQRC_DURABILITY_NOT_ALTERABLE
MQSD	Bestimmungsortoptionen	Ja	--
MQSD	Registrierungsoptionen	Ja (siehe Hinweis „1” auf Seite 559)	MQRC_GROUPING_NOT_ALTERABLE, wenn Sie versuchen, MQSO_GROUP_SUB zu ändern
MQSD	Veröffentlichungsoptionen	Ja (siehe Hinweis „2” auf Seite 559)	--
MQSD	Platzhalteroptionen	Nein	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Sonstige Optionen	Nein (siehe Hinweis „3” auf Seite 559)	--
MQSD	ObjectName	Nein	MQRC_TOPIC_NOT_ALTERABLE
MQSD	AlternateUserId	Nein (siehe Hinweis „4” auf Seite 559)	--
MQSD	AlternateSecurityId	Nein (siehe Hinweis „4” auf Seite 559)	--
MQSD	SubExpiry	Ja	--
MQSD	ObjectString	Nein	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	Nein (siehe Hinweis „5” auf Seite 559)	--
MQSD	SubUserData	Ja	--
MQSD	SubCorrelId	Ja (siehe Hinweis „6” auf Seite 559)	MQRC_GROUPING_NOT_ALTERABLE bei Vorliegen in einer gruppierten Subskription
MQSD	PubPriority	Ja	--

Tabelle 547. Attribute in MQSD und MQSUB, die geändert werden können (Forts.)

Datentypdeskriptor oder Funktionsaufruf	Name des Felds	Kann dieses Attribut mit MQSO_ALTER geändert werden	Ursachencode
MQSD	PubAccountingToken	Ja	--
MQSD	PubApplIdentityData	Ja	--
MQSD	SubLevel	Nein	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	Hobj	Ja (siehe Hinweis „6" auf Seite 559)	MQRC_GROUPING_NOT_ALTERABLE bei Vorliegen in einer gruppierten Subskription

Anmerkungen:

1. MQSO_GROUP_SUB kann nicht geändert werden.
2. MQSO_NEW_PUBLICATIONS_ONLY kann nicht geändert werden, da es nicht Bestandteil der Subskription ist
3. Diese Optionen sind kein Bestandteil der Subskription
4. Dieses Attribut ist kein Bestandteil der Subskription
5. Dieses Attribut ist die Identität der Subskription, die geändert wird
6. Änderbar, wenn nicht Bestandteil einer gruppierten Subskription (MQSO_GROUP_SUB)

Lebensdaueroptionen: Die folgenden Optionen steuern die Lebensdauer der Subskription. Es kann nur eine dieser Optionen angegeben werden. Wenn Sie eine vorhandene Subskription mit der Option MQSO_ALTER ändern, können Sie die Lebensdauer der Subskription nicht ändern. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird die entsprechende Dauerhaltbarkeitsoption festgelegt.

MQSO_DURABLE
Fordert an, dass die Subskription dieses Themas bestehen bleibt, bis sie explizit durch MQCLOSE mit der Option MQCO_REMOVE_SUB entfernt wird. Wird diese Subskription nicht explizit entfernt, wird sie auch beibehalten, wenn die Verbindung dieser Anwendung zum Warteschlangenmanager geschlossen wurde.

Wenn eine permanente Subskription für ein Thema angefordert wird, das gemäß seiner Definition keine permanenten Subskriptionen zulässt, schlägt der Aufruf mit MQRC_DURABILITY_NOT_ALLOWED fehl.

MQSO_NON_DURABLE
Fordert an, dass die Subskription für dieses Thema entfernt wird, wenn die Verbindung der Anwendung zum Warteschlangenmanager geschlossen wurde, sofern sie noch nicht explizit entfernt wurde. MQSO_NON_DURABLE ist das Gegenstück zur Option MQSO_DURABLE und wird zur Unterstützung der Programmdokumentation definiert. Es handelt sich dabei um den Standardwert, wenn nichts anderes angegeben ist.

Zieloptionen: Die folgende Option legt das Ziel fest, an das die Veröffentlichungen für ein abonniertes Thema gesendet werden. Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann das für Veröffentlichungen für die Subskription verwendete Ziel geändert werden. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird diese Option bei Bedarf festgelegt.

MQSO_MANAGED

Fordert an, dass das Ziel, zu dem die Veröffentlichungen gesendet werden, vom Warteschlangenmanager verwaltet wird.

Die in *Hobj* zurückgegebene Kennung stellt eine vom Warteschlangenmanager verwaltete Warteschlange dar und ist für die Verwendung mit den nachfolgenden Aufrufen MQGET, MQCB, MQINQ oder MQCLOSE vorgesehen.

Eine von einem vorherigen MQSUB-Aufruf zurückgegebene Objektkennung kann nicht im Parameter *Hobj* bereitgestellt werden, wenn MQSO_MANAGED nicht angegeben ist.

MQSO_NO_MULTICAST

Fordert an, dass das Ziel, an das die Veröffentlichungen gesendet werden, keine Multicastgruppenadresse ist. Diese Option ist nur gültig, wenn sie mit der Option MQSO_MANAGED kombiniert wird. Wenn

eine Kennung für eine Warteschlange im Parameter *Hobj* bereitgestellt wird, kann Multicasting nicht für diese Subskription verwendet werden und die Option ist nicht gültig.

Wenn das Thema über die Einstellung MCAST(ONLY) so definiert ist, dass es nur Multicastsubskriptionen zulässt, schlägt der Aufruf mit dem Ursachencode MQRC_MULTICAST_REQUIRED fehl.

Bereichsoption: Die folgende Option steuert den Geltungsbereich der Subskription, die erstellt wird. Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann die Bereichsoption dieser Subskription nicht geändert werden. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO-RESUME wird die entsprechende Bereichsoption festgelegt.

MQSO_SCOPE_QMGR
Diese Subskription wird nur auf dem lokalen Warteschlangenmanager erstellt. Es wird keine Proxy-Subskription an andere Warteschlangenmanager im Netz verteilt. Nur Veröffentlichungen, die auf diesem Warteschlangenmanager publiziert werden, werden an diesen Subskribenten gesendet. Dies überschreibt das mit dem Themenattribut SUBSCOPE festgelegte Verhalten.

Anmerkung: Ist diese Option nicht angegeben, wird der Subskriptionsbereich durch das Themenattribut SUBSCOPE festgelegt.

Registrierungsoptionen: Die folgenden Optionen steuern die Details der Registrierung, die auf dem Warteschlangenmanager für diese Subskription vorgenommen wird. Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, können diese Registrierungsoptionen geändert werden. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME werden die entsprechenden Registrierungsoptionen festgelegt.

MQSO_GROUP_SUB

Diese Subskription soll unter Verwendung derselben Warteschlange und Angabe derselben Korrelations-ID mit anderen Subskriptionen derselben SubLevel gruppiert werden, sodass alle Veröffentlichungen zu Themen, die dazu führen würden, dass mehrere Veröffentlichungsnachrichten an die Gruppe der Subskriptionen ausgegeben werden, da eine überlappende Gruppe von Themenzeichenfolgen verwendet wird, dazu führen, dass nur eine Nachricht an die Warteschlange übergeben wird. Wenn diese Option nicht verwendet wird, wird jede übereinstimmende eindeutige Subskription (durch SubName identifiziert), mit einer Kopie der Veröffentlichung bereitgestellt. Dies könnte bedeuten, dass mehrere Kopien der Veröffentlichung in der von einer Reihe von Subskriptionen gemeinsam genutzten Warteschlange abgelegt werden.

Nur die wichtigste Subskription in der Gruppe erhält eine Kopie der Veröffentlichung. Die wichtigste Subskription basiert auf dem vollständigen Themennamen bis zu dem Punkt, an dem ein Platzhalter gefunden wird. Wenn eine Mischung aus Platzhalterschemata in der Gruppe verwendet wird, ist nur die Position des Platzhalters von Bedeutung. Es wird empfohlen, innerhalb einer Gruppe von Subskriptionen, die dieselbe Warteschlange benutzen, keine unterschiedlichen Platzhalterschemata zu kombinieren.

Wenn eine neue gruppierte Subskription erstellt wird, muss diese einen eindeutigen SubName aufweisen. Stimmt dieser jedoch mit dem vollständigen Themennamen einer vorhandenen Subskription in der Gruppe überein, schlägt der Aufruf mit MQRC_DUPLICATE_GROUP_SUB fehl.

Wenn die wichtigste Subskription in der Gruppe auch MQSO_NOT_OWN_PUBS angibt und dies eine Veröffentlichung aus derselben Anwendung ist, wird keine Veröffentlichung an die Warteschlange übergeben.

Wenn eine mit dieser Option erstellte Subskription geändert wird, können die Felder, die die Gruppierung implizieren, das Hobj im Aufruf MQSUB (das die Warteschlange und den Namen des Warteschlangenmanagers darstellt), und die SubCorrelId nicht geändert werden. Der Versuch, diese Werte zu ändern, führt dazu, dass der Aufruf mit MQRC_GROUPING_NOT_ALTERABLE fehlschlägt.

Diese Option muss mit MQSO_SET_CORREL_ID mit einer SubCorrelId kombiniert werden, die nicht auf MQCI_NONE gesetzt ist. Die Option kann nicht mit MQSO_MANAGED kombiniert werden.

MQSO_ANY_USERID

Wenn MQSO_ANY_USERID angegeben ist, ist die Identität des Subskribenten nicht auf eine einzelne Benutzer-ID eingeschränkt. Dadurch kann jeder Benutzer die Subskription ändern oder fortsetzen,

sofern er über die entsprechende Berechtigung verfügt. Die Subskription kann jeweils nur einem einzelnen Benutzer gehören. Ein Versuch, die Verwendung einer Subskription fortzusetzen, die derzeit von einer anderen Anwendung verwendet wird, führt dazu, dass der Aufruf mit MQRC_SUBSCRIPTION_IN_USE fehlschlägt.

Um diese Option einer vorhandenen Subskription hinzuzufügen, muss der Aufruf MQSUB (unter Verwendung von MQSO_ALTER) aus derselben Benutzer-ID stammen wie die ursprüngliche Subskription.

Wenn sich ein MQSUB-Aufruf auf eine vorhandene Subskription bezieht, für die MQSO_ANY_USERID festgelegt ist, und die Benutzer-ID von der ursprünglichen Subskription abweicht, ist der Aufruf nur erfolgreich, wenn die neue Benutzer-ID über die Berechtigung verfügt, das Thema zu abonnieren. Bei einer erfolgreichen Ausführung werden zukünftige Veröffentlichungen an diesen Subskribenten in die Warteschlange des Subskribenten eingereicht, wobei die neue Benutzer-ID in der Veröffentlichungsnachricht festgelegt wird.

Geben Sie nicht MQSO_ANY_USERID zusammen mit MQSO_FIXED_USERID an. Ist keine der beiden Optionen angegeben, ist MQSO_FIXED_USERID der Standardwert.

MQSO_FIXED_USERID

Wenn MQSO_FIXED_USERID angegeben wird, kann die Subskription nur von der letzten Benutzer-ID zum Ändern der Subskription geändert oder fortgesetzt werden. Wurde die Subskription nicht geändert, ist es die Benutzer-ID, von der die Subskription erstellt wurde.

Wenn sich ein MQSUB-Verb auf eine vorhandene Subskription bezieht, für die MQSO_ANY_USERID festgelegt ist, und die Subskription mit MQSO_ALTER so ändert, dass die Option MQSO_FIXED_USERID verwendet wird, ist die Benutzer-ID der Subskription jetzt auf diese neue Benutzer-ID festgelegt. Der Aufruf ist nur erfolgreich, wenn die neue Benutzer-ID befugt ist, das Thema zu abonnieren.

Wenn eine andere Benutzer-ID als die, die als Eigner einer Subskription dokumentiert ist, versucht, eine Subskription mit MQSO_FIXED_USERID fortzusetzen oder zu ändern, schlägt der Aufruf mit MQRC_IDENTITY_MISMATCH fehl. Die Benutzer-ID, die Eigner einer Subskription ist, kann mit dem Befehl DISPLAY SBSTATUS angezeigt werden.

Geben Sie nicht MQSO_ANY_USERID zusammen mit MQSO_FIXED_USERID an. Ist keine der beiden Optionen angegeben, ist MQSO_FIXED_USERID der Standardwert.

Veröffentlichungsoptionen: Die folgenden Optionen steuern, wie Veröffentlichungen an diesen Subskribenten gesendet werden. Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, können diese Veröffentlichungsoptionen geändert werden.

MQSO_NOT_OWN_PUBS
Über diese Option wird dem Broker mitgeteilt, dass die Anwendung keine ihre eigenen Veröffentlichungen sehen will. Es ist festgelegt, dass Veröffentlichungen aus derselben Anwendung stammen, wenn die Verbindungskennungen identisch sind. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird diese Option bei Bedarf festgelegt.

MQSO_NEW_PUBLICATIONS_ONLY
Es werden keine aktuellen ständigen Veröffentlichungen gesendet, wenn diese Subskription erstellt wird, sondern nur neue Veröffentlichungen. Diese Option gilt nur, wenn MQSO_CREATE angegeben ist. Alle nachfolgenden Änderungen an einer Subskription ändern die Übertragung von Veröffentlichungen nicht, sodass alle zu einem Thema aufbewahrten Veröffentlichungen bereits als neue Veröffentlichungen an den Subskribenten gesendet wurden.

Wenn diese Option ohne MQSO_CREATE angegeben wird, schlägt der Aufruf mit MQRC_OPTIONS_ERROR fehl. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird diese Option auch dann nicht festgelegt, wenn die Subskription mit dieser Option erstellt wurde.

Wird diese Option nicht verwendet, werden zuvor beibehaltene Nachrichten zu der angegebenen Zielwarteschlange gesendet. Wenn diese Aktion aufgrund eines Fehlers (MQRC_RETAINED_MSG_Q_ERROR oder MQRC_RETAINED_NOT_DELIVERED) fehlschlägt, schlägt die Erstellung der Subskription fehl.

MQSO_PUBLICATIONS_ON_REQUEST
Das Festlegen dieser Option gibt an, dass der Subskribent Informationen gesondert anfordert, wenn diese benötigt werden. Der Warteschlangenmanager sendet keine Nachrichten an den Abonnenten, die dieser nicht angefordert hat. Die ständige Veröffentlichung (oder auch mehrere Veröffentlichungen, wenn ein Platzhalter im Thema angegeben ist) wird immer dann zum Abonnenten gesendet, wenn ein MQSUBRQ-Aufruf mit der Hsub-Kennung aus einem vorherigen MQSUB-Aufruf durchgeführt wird. Es werden keine Veröffentlichungen gesendet, wenn diese Option für den MQSUB-Aufruf angegeben ist. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird diese Option bei Bedarf festgelegt.

Diese Option ist in Verbindung mit einer SubLevel größer als 1 nicht gültig.

Vorausleseoptionen: Die folgenden Optionen steuern, ob nicht persistente Nachrichten an eine Anwendung gesendet werden, bevor die Anwendung sie anfordert.

MQSO_READ_AHEAD_AS_Q_DEF
Wenn der Aufruf MQSUB eine verwaltete Kennung verwendet, legt das Standardvorausleseattribut der Modellwarteschlange, die dem subskribierten Thema zugeordnet ist, fest, ob Nachrichten an die Anwendung gesendet werden, bevor die Anwendung sie anfordert.

Dies ist der Standardwert.

MQSO_NO_READ_AHEAD
Wenn der Aufruf MQSUB eine verwaltete Kennung verwendet, werden Nachrichten nicht an die Anwendung gesendet, bevor die Anwendung sie anfordert.

MQSO_READ_AHEAD
Wenn der Aufruf MQSUB eine verwaltete Kennung verwendet, werden Nachrichten möglicherweise an die Anwendung gesendet, bevor die Anwendung sie anfordert.

Anmerkung:

Für Vorausleseoptionen gelten folgende Hinweise:

1. Es kann nur eine dieser Optionen angegeben werden. Wenn sowohl MQOO_READ_AHEAD als auch MQOO_NO_READ_AHEAD angegeben werden, wird der Ursachencode MQRC_OPTIONS_ERROR zurückgegeben. Diese Optionen gelten nur, wenn MQSO_MANAGED angegeben ist.
2. Sie gelten nicht für MQSUB, wenn eine Warteschlange übergeben wird, die zuvor geöffnet wurde. Das Vorauslesen wird möglicherweise nicht aktiviert, wenn es angefordert wird. Die im ersten MQGET-Aufruf verwendeten MQGET-Optionen verhindern unter Umständen, dass das Vorauslesen aktiviert wird. Ebenso wird das Vorauslesen inaktiviert, wenn der Client eine Verbindung zu einem Warteschlangenmanager herstellt, auf dem das Vorauslesen nicht unterstützt wird. Wenn die Anwendung nicht als WebSphere MQ-Client ausgeführt wird, werden diese Optionen ignoriert.

Platzhalteroptionen: Die folgenden Optionen steuern, wie Platzhalter in der Zeichenfolge interpretiert werden, die im Feld ObjectString des MQSD bereitgestellt wird. Es kann nur eine dieser Optionen angegeben werden. Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, können diese Platzhalteroptionen nicht geändert werden. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird die entsprechende Platzhalteroption festgelegt.

MQSO_WILDCARD_CHAR
Platzhalter können nur für Zeichen innerhalb der Themenzeichenfolge verwendet werden.

Das durch MQSO_WILDCARD_CHAR definierte Verhalten ist in der folgenden Tabelle dargestellt.

Sonderzeichen	**Verhalten**
Schrägstrich (/)	Keine Signifikanz, nur ein anderes Zeichen
Stern (*)	Platzhalter, null oder mehr Zeichen
Fragezeichen (?)	Platzhalter, 1 Zeichen

Sonderzeichen	Verhalten
Prozentzeichen (%)	Escapezeichen zum Zulassen, dass die Zeichen (*), (?) oder (%) in einer Zeichenfolge verwendet werden und nicht als Sonderzeichen interpretiert werden, z. B. (%*), (%?) oder (%%).

Die Veröffentlichung zu folgendem Thema:

```
/level0/level1/level2/level3/level4
```

stimmt beispielsweise mit Subskribenten überein, die die folgenden Themen verwenden:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Anmerkung: Diese Verwendung von Platzhaltern liefert exakt die Bedeutung, die in WebSphere MQ V6 und WebSphere MB V6 bei Verwendung formatierter MQRFH1-Nachrichten für Publish/Subscribe bereitgestellt wird. Es wird empfohlen, dies nicht für neu erstellte Anwendung zu verwenden, sondern nur für Anwendungen, die zuvor mit dieser Version ausgeführt wurden und nicht geändert wurden, sodass sie das in MQSO_WILDCARD_TOPIC beschriebene Standardplatzhalterverhalten verwenden.

MQSO_WILDCARD_TOPIC

Platzhalter wirken sich nur auf Themenelemente innerhalb der Themenzeichenfolge aus. Dies ist das Standardverhalten, wenn "Keine" ausgewählt wird.

Das für MQSO_WILDCARD_TOPIC erforderliche Verhalten ist in der folgenden Tabelle dargestellt:

Sonderzeichen	Verhalten
(/)	Trennzeichen auf Themenebene
Nummernzeichen (#)	Platzhalter: mehrere Themenebenen
Pluszeichen (+)	Platzhalter: eine Themenebene

Anmerkungen:

Die Zeichen (+) und (#) werden nicht als Platzhalter behandelt, wenn sie innerhalb einer Themenebene mit anderen Zeichen (einschließlich sich selbst) gemischt werden. In der folgenden Zeichenfolge werden die Zeichen (#) und (+) als normale Zeichen behandelt.

```
level0/level1/#+/level3/level#
```

Die Veröffentlichung zu folgendem Thema:

```
/level0/level1/level2/level3/level4
```

stimmt beispielsweise mit Subskribenten überein, die die folgenden Themen verwenden:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Anmerkung: Diese Verwendung von Platzhaltern liefert die Bedeutung, die in WebSphere Message Broker Version 6 bei Verwendung von formatierten MQRFH2-Nachrichten für Publish/Subscribe bereitgestellt wird.

Sonstige Optionen: Die folgenden Optionen steuern die Art und Weise, auf die der API-Aufruf und nicht die Subskription ausgegeben wird. Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von

MQSO_RESUME bleiben diese Optionen unverändert. Weitere Informationen finden Sie in „AlternateUserId (MQCHAR12)" auf Seite 554.

MQSO_ALTERNATE_USER_AUTHORITY

Das Feld AlternateUserId enthält eine Benutzer-ID zur Überprüfung dieses MQSUB-Aufrufs. Der Aufruf kann nur erfolgreich ausgeführt werden, wenn diese AlternateUserId berechtigt ist, das Objekt mit den angegebenen Zugriffsoptionen zu öffnen, und zwar unabhängig davon, ob die Benutzer-ID, unter der die Anwendung ausgeführt wird, dazu berechtigt ist.

MQSO_SET_CORREL_ID

Die Subskription soll die Korrelations-ID verwenden, die im Feld *SubCorrelId* angegeben ist. Wenn diese Option nicht angegeben ist, erstellt der Warteschlangenmanager zum Zeitpunkt der Subskription automatisch eine Korrelations-ID, die im Feld *SubCorrelId* an die Anwendung zurückgegeben wird. Weitere Informationen finden Sie unter „SubCorrelId (MQBYTE24)" auf Seite 567.

Diese Option kann nicht mit MQSO_MANAGED kombiniert werden.

MQSO_SET_IDENTITY_CONTEXT

Die Subskription soll den im Feld *PubAccountingToken* angegebenen Abrechnungstoken und die im Feld *PubApplIdentityData* angegebenen Abrechnungsidentitätsdaten verwenden.

Wenn diese Option angegeben wird, wird dieselbe Berechtigungsprüfung ausgeführt, als wäre der Zugriff auf die Zielwarteschlange über einen MQOPEN-Aufruf mit MQOO_SET_IDENTITY_CONTEXT erfolgt. Dies gilt nicht für den Fall, dass die Option MQSO_MANAGED ebenfalls verwendet wird. In diesem Fall erfolgt keine Berechtigungsprüfung in der Zielwarteschlange.

Wenn diese Option nicht angegeben wird, sind den Veröffentlichungen, die an diesen Subskribenten gesendet werden, folgende Standardkontextinformationen zugeordnet:

Feld im MQMD	Verwendeter Wert
UserIdentifier	Dies ist die Benutzer-ID, die zum Zeitpunkt der Erstellung der Subskription mit dieser verknüpft war.
AccountingToken	Wird, falls möglich, anhand der Umgebung ermittelt; wird andernfalls auf MQACT_NONE gesetzt.
ApplIdentityData	Wird auf Leerzeichen gesetzt

Diese Option ist nur mit MQSO_CREATE und MQSO_ALTER gültig. Wird die Option mit MQSO_RESUME verwendet, werden die Felder *PubAccountingToken* und *PubApplIdentityData* ignoriert, sodass diese Option keine Auswirkungen hat.

Wenn eine Subskription ohne Verwendung dieser Option geändert wurde und die Subskription zuvor Identitätskontextinformationen bereitgestellt hat, werden Standardkontextinformationen für die geänderte Subskription generiert.

Wenn eine Subskription, die zulässt, dass verschiedene Benutzer-IDs sie mit der Option MQSO_ANY_USERID verwenden, von einer anderen Benutzer-ID fortgesetzt wird, wird ein Standardidentitätskontext für die neue Benutzer-ID generiert, die jetzt Eigner der Subskription ist. Alle nachfolgenden Veröffentlichungen werden mit dem neuen Identitätskontext bereitgestellt.

MQSO_FAIL_IF_QUIESCING

Der MQSUB-Aufruf schlägt fehl, wenn der Warteschlangenmanager sich im Quiescestatus befindet. In einer unter z/OS ausgeführten CICS- oder IMS-Anwendung bewirkt diese Option auch das Fehlschlagen des MQSUB-Aufrufs wenn sich die Verbindung im Quiescestatus befindet.

PubAccountingToken (MQBYTE32)

Dies ist der Wert im Feld *AccountingToken* des Nachrichtendeskriptors aller Veröffentlichungsnachrichten, die dieser Subskription entsprechen. *AccountingToken* ist Teil des Identitätskontexts der

Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext. Weitere Informationen zum Feld *AccountingToken* im Nachrichtendeskriptor finden Sie unter „AccountingToken (MQBYTE32)" auf Seite 402.

Sie können den folgenden Sonderwert für das Feld *PubAccountingToken* verwenden:

MQACT_NONE

Es ist kein Abrechnungstoken angegeben.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQACT_NONE_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQACT_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Wenn die Option MQSO_SET_IDENTITY_CONTEXT nicht angegeben ist, wird der Abrechnungstoken vom Warteschlangenmanager als Standardkontextinformation generiert und dieses Feld ist ein Ausgabefeld, das den *AccountingToken* enthält, der in jeder für diese Subskription veröffentlichten Nachricht festgelegt wird.

Wenn die Option MQSO_SET_IDENTITY_CONTEXT angegeben ist, wird der Abrechnungstoken vom Benutzer generiert. Dieses Feld ist ein Eingabefeld, das den *AccountingToken* enthält, der in jeder Veröffentlichung für diese Subskription festzulegen ist.

Die Länge dieses Felds wird durch MQ_ACCOUNTING_TOKEN_LENGTH angegeben. Der Anfangswert dieses Felds ist MQACT_NONE.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann der Wert von *AccountingToken* aller zukünftigen Veröffentlichungsnachrichten geändert werden.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird dieses Feld auf den aktuellen *AccountingToken* gesetzt, der für die Subskription verwendet wird.

PubApplIdentityData (MQCHAR32)

Dies ist der Wert im Feld *ApplIdentityData* des Nachrichtendeskriptors aller Veröffentlichungsnachrichten, die dieser Subskription entsprechen. *ApplIdentityData* ist Teil des Identitätskontexts der Nachricht. Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext. Weitere Informationen zum Feld *ApplIdentityData* im Nachrichtendeskriptor finden Sie unter „ApplIdentityData (MQCHAR32)" auf Seite 404.

Wenn die Option MQSO_SET_IDENTITY_CONTEXT nicht angegeben ist, ist das Feld *ApplIdentityData*, das in jeder für diese Subskription veröffentlichten Nachricht als Standardkontextinformationen festgelegt wird, leer.

Wenn die Option MQSO_SET_IDENTITY_CONTEXT angegeben ist, werden die *PubApplIdentityData* vom Benutzer generiert. Dieses Feld ist ein Eingabefeld, das die *ApplIdentityData* enthält, die in jeder Veröffentlichung für diese Subskription festzulegen sind.

Die Länge dieses Felds wird durch MQ_APPL_IDENTITY_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 32 Leerzeichen in anderen Programmiersprachen.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, können die *ApplIdentityData* aller zukünftigen Veröffentlichungsnachrichten geändert werden.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird dieses Feld auf die aktuellen *ApplIdentityData* gesetzt, die für die Subskription verwendet werden.

PubPriority (MQLONG)

Dies ist der Wert im Feld *Priority* des Nachrichtendeskriptors aller Veröffentlichungsnachrichten, die dieser Subskription entsprechen. Weitere Informationen zum Feld *Priority* im Nachrichtendeskriptor finden Sie unter „Priority (MQLONG)" auf Seite 430.

Der Wert muss größer oder gleich Null sein. Null steht für die niedrigste Priorität. Die folgenden besonderen Werte können ebenfalls verwendet werden:

MQPRI_PRIORITY_AS_Q_DEF

Wenn eine Subskriptionswarteschlange im Feld *Hobj* des Aufrufs MQSUB bereitgestellt wird und keine verwaltete Kennung ist, wird die Priorität für die Nachricht dem Attribut *DefPriority* dieser Warteschlange entnommen. Wenn es sich bei der Warteschlange um eine Clusterwarteschlange handelt oder es mehrere Definitionen im Auflösungspfad des Warteschlangenamens gibt, wird die Priorität bestimmt, wenn die Veröffentlichungsnachricht, wie für „Priority (MQLONG)” auf Seite 430 beschrieben, in die Warteschlange eingereiht wird.

Wenn der MQSUB-Aufruf eine verwaltete Kennung verwendet, wird die Priorität für die Nachricht dem Attribut *DefPriority* der Modellwarteschlange entnommen, die mit dem abonnierten Thema verknüpft ist.

MQPRI_PRIORITY_AS_PUBLISHED

Die Priorität für die Nachricht ist die Priorität der ursprünglichen Veröffentlichung. Dies ist der Anfangswert des Felds.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann die *Priority* aller zukünftigen Veröffentlichungsnachrichten geändert werden.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird dieses Feld auf die aktuelle Priorität gesetzt, die für die Subskription verwendet wird.

ResObjectString (MQCHARV)
Dies ist der ausgeschriebene Objektname, nachdem der Warteschlangenmanager den in *ObjectName* bereitgestellten Namen aufgelöst hat.

Wenn der ausgeschriebene Objektname in *ObjectString* bereitgestellt wird und im Feld *ObjectName* keine Angaben gemacht werden, ist der in diesem Feld zurückgegebene Wert mit dem in *ObjectString* bereitgestellten Wert identisch.

Wenn das Feld übergangen wird (d. h. ResObjectString.VSBufSize ist null), wird die *ResObjectString* nicht zurückgegeben. Stattdessen wird die Länge in ResObjectString.VSLength zurückgegeben. Wenn die Länge kürzer ist als die vollständige ResObjectString, wird sie abgeschnitten und gibt so viele der Zeichen ganz rechts zurück wie in die bereitgestellte Länge passen.

Wenn *ResObjectString* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der MQCHARV-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_RES_OBJECT_STRING_ERROR fehl.

SelectionString (MQCHARV)
Dies ist die Zeichenfolge, die verwendet wird, um die Auswahlkriterien anzugeben, die beim Abonnieren von Nachrichten von einem Thema verwendet werden.

Dieses Feld mit variabler Länge wird bei der Ausgabe eines MQSUB-Aufrufs mit der Option MQSO_RESUME zurückgegeben, wenn ein Puffer bereitgestellt wird und außerdem in "VSBufSize" eine positive Puffergröße angegeben ist. Wird beim Aufruf kein Puffer bereitgestellt, wird im Feld VSLength von MQCHARV nur die Länge der Auswahlzeichenfolge zurückgegeben. Ist der bereitgestellte Puffer kleiner als der für die Rückgabe des Feldes erforderliche Speicherplatz, werden nur VSBufSize-Bytes im bereitgestellten Puffer zurückgegeben.

Wenn *SelectionString* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der „MQCHARV - Zeichenfolge variabler Länge” auf Seite 277-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_SELECTION_STRING_ERROR fehl.

Die Verwendung von SelectionString wird unter Selektoren beschrieben.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQSD_STRUC_ID
ID für die Struktur des Subskriptionsdeskriptors.

Für die Programmiersprache C ist auch die Konstante MQSD_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQSD_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSD_STRUC_ID.

SubCorrelId (MQBYTE24)
Dieses Feld enthält eine Korrelations-ID, die allen Veröffentlichungen, die mit dieser Subskription übereinstimmen, gemeinsam ist.

Achtung: Eine Korrelations-ID kann nur zwischen Warteschlangenmanagern in einem Publish/Subscribe-Cluster übergeben werden, nicht in einer Hierarchie.

Alle gesendeten Veröffentlichungen, die mit dieser Subskription übereinstimmen, enthalten diese Korrelations-ID im Nachrichtendeskriptor. Wenn mehrere Subskriptionen ihre Veröffentlichungen aus derselben Warteschlange abrufen, ermöglicht die Verwendung von MQGET nach Korrelations-ID nur das Abrufen von Veröffentlichungen für eine bestimmte Subskription. Diese Korrelations-ID kann entweder vom Warteschlangenmanager oder vom Benutzer generiert werden.

Wenn die Option MQSO_SET_CORREL_ID nicht angegeben ist, wird die Korrelations-ID vom Warteschlangenmanager generiert. Dieses Feld ist ein Ausgabefeld, das die Korrelations-ID enthält, die in jeder für diese Subskription veröffentlichten Nachricht festgelegt wird. Die generierte Korrelations-ID besteht aus einer Produkt-ID mit 4 Byte (AMQX oder CSQM in ASCII oder EBCDIC), gefolgt von einer produktspezifischen Implementierung einer eindeutigen Zeichenfolge.

Wenn die Option MQSO_SET_CORREL_ID angegeben ist, wird die Korrelations-ID vom Benutzer generiert. Dieses Feld ist ein Eingabefeld, das die Korrelations-ID enthält, die in jeder Veröffentlichung für diese Subskription festzulegen ist. Wenn das Feld in diesem Fall die Option MQCI_NONE enthält, ist die Korrelations-ID, die in jeder für diese Subskription veröffentlichten Nachricht festgelegt wird, die Korrelations-ID, die beim ursprünglichen Einreihen dieser Nachricht in die Warteschlange erstellt wurde.

Wenn die Option MQSO_GROUP_SUB angegeben ist und die angegebene Korrelations-ID mit einer vorhandenen gruppierten Subskription übereinstimmt, die dieselbe Warteschlange und eine überlappende Themenzeichenfolge verwendet, erhält nur die höchstwertige Subskription in der Gruppe eine Kopie der Veröffentlichung.

Die Länge dieses Felds wird durch MQ_CORREL_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist MQCI_NONE.

Wenn Sie eine vorhandene Subskription mit der Option MQSO_ALTER ändern und dieses Feld ein Eingabefeld ist, kann die Korrelations-ID der Subskription geändert werden, wenn die Subskription keine gruppierte Subskription ist, d. h. dass sie mit der Option MQSO_GROUP_SUB erstellt wurde. In diesem Fall kann die Korrelations-ID der Subskription nicht geändert werden.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird dieses Feld auf die aktuelle Korrelations-ID für die Subskription gesetzt.

SubExpiry (MQLONG)

Dies ist die Zeit, nach der die Subskription abläuft, ausgedrückt in Zehntelsekunden. Wenn dieses Intervall verstrichen ist, stimmen keine Veröffentlichungen mehr mit der Subskription überein. Sobald eine Subskription abläuft, werden keine Veröffentlichungen mehr an die Warteschlange gesendet. Die bereits darin enthaltenen Veröffentlichungen sind davon jedoch nicht betroffen. *SubExpiry* hat keine Auswirkungen auf den Ablauf von Veröffentlichungen.

Der folgende Sonderwert wird erkannt:

MQEI_UNLIMITED
Die Subskription hat eine unbegrenzte Ablaufzeit.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann der Ablauf der Subskription geändert werden.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung der Option MQSO_RESUME wird dieses Feld auf die ursprüngliche Ablaufzeit der Subskription gesetzt, nicht auf die verbleibende Ablaufzeit.

SubLevel (MQLONG)
Dies ist die mit der Subskription verknüpfte Ebene. Veröffentlichungen werden nur an diese Subskription übermittelt, wenn sie sich in der Gruppe der Subskriptionen befindet, bei der der höchste Wert für 'SubLevel' kleiner-gleich dem Wert für 'PubLevel' ist, der zur Veröffentlichungszeit verwendet wurde. Wenn eine Veröffentlichung jedoch beibehalten wurde, ist sie für Abonnenten auf höheren Ebenen nicht mehr verfügbar, da sie auf PubLevel 1 erneut veröffentlicht wird.

Der Wert muss im Bereich zwischen null und 9 liegen. Null ist die niedrigste Stufe.

Der Anfangswert dieses Felds ist 1.

Weitere Informationen finden Sie unter Veröffentlichungen abfängt .

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann die SubLevel nicht geändert werden.

Die Kombination einer SubLevel mit einem Wert größer als 1 mit der Option MQSO_PUBLICATIONS_ON_REQUEST ist nicht zulässig.

Bei der Rückgabe von einem MQSUB-Aufruf unter Verwendung von MQSO_RESUME wird dieses Feld auf die aktuelle Ebene gesetzt, die für die Subskription verwendet wird.

SubUserData (MQCHARV)

Dies gibt die Subskriptionsbenutzerdaten an. Die Daten, die bei Subskription in diesem Feld angegeben werden, sind als Nachrichteneigenschaft 'MQSubUserData' jeder Veröffentlichung enthalten, die an diese Subskription gesendet wird.

Die maximale Länge von *SubUserData* beträgt 10240.

Wenn *SubUserData* nicht ordnungsgemäß gemäß der Beschreibung zur Verwendung der MQCHARV-Struktur angegeben wird oder die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_SUB_USER_DATA_ERROR fehl.

Dies ist ein Eingabefeld. Die Anfangswerte der Felder in dieser Struktur entsprechen denen der MQCHARV-Struktur.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, können die Subskriptionsbenutzerdaten geändert werden.

Dieses Feld mit variablen Längen wird bei der Ausgabe eines MQSUB-Aufrufs mit der Option MQSO_RESUME zurückgegeben, wenn ein Puffer bereitgestellt wird und in *VSBufLen* eine positive Puffergröße angegeben ist. Wird im Aufruf kein Puffer bereitgestellt, wird nur die Länge der Subskriptionsbenutzerdaten im Feld *VSLength* der MQCHARV-Struktur zurückgegeben. Ist der bereitgestellte Puffer kleiner als der für die Rückgabe des Felds erforderliche Speicherplatz, werden nur *VSBufLen*-Bytes im bereitgestellten Puffer zurückgegeben.

SubName (MQCHARV)

Dies gibt den Subskriptionsnamen an. Dieses Feld ist nur erforderlich, wenn *Options* die Option MQSO_DURABLE angibt. Wird das Feld bereitgestellt, wird es vom Warteschlangenmanager jedoch auch für MQSO_NON_DURABLE verwendet.

Wenn das Feld angegeben wird, muss *SubName* im Warteschlangenmanager eindeutig sein, da dies die Methode ist, die zum Angeben der Subskription verwendet wird.

Die maximale Länge von *SubName* beträgt 10240.

Dieses Feld dient zu zwei Zwecken. Für eine MQSO_DURABLE-Subskription verwenden Sie dieses Feld, um eine Subskription zu kennzeichnen, sodass Sie diese nach der Erstellung fortsetzen können, wenn Sie die Kennung für die Subskription geschlossen haben (mit der Option MQCO_KEEP_SUB) oder die Verbindung zum Warteschlangenmanager getrennt wurde. Dies erfolgt mit dem Aufruf MQSUB mit der Option MQSO_RESUME. Dies wird auch in der Verwaltungsansicht von Subskriptionen im Feld SUBNAME in DISPLAY SBSTATUS angezeigt.

Wenn *SubName* gemäß der Beschreibung zur Verwendung der MQCHARV -Struktur falsch angegeben wird, wird es weggelassen, wenn es erforderlich ist (d. h. *SubName*.*VSLength* ist null) oder wenn er die maximale Länge überschreitet, schlägt der Aufruf mit dem Ursachencode MQRC_SUB_NAME_ERROR fehl.

Dies ist ein Eingabefeld. Die Anfangswerte der Felder in dieser Struktur entsprechen denen der MQCHARV-Struktur.

Wenn eine vorhandene Subskription mit der Option MQSO_ALTER geändert wird, kann der Name der Subskription nicht geändert werden, da dies das Kennzeichnungsfeld ist, das zum Suchen der referenzierten Subskription verwendet wird. Es wird bei der Ausgabe aus einem MQSUB-Aufruf mit der Option MQSO_RESUME nicht geändert.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQSD_VERSION_1
Struktur des Subskriptionsdeskriptors der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQSD_CURRENT_VERSION
Aktuelle Version der Struktur des Subskriptionsdeskriptors.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSD_VERSION_1.

Verwenden von Themenzeichenfolgen

Ein Thema wird aus dem in einem Themenobjekt angegebenen Unterthema und einem von einer Anwendung bereitgestellten Unterthema erstellt. Sie können das Unterthema als Themennamen verwenden oder die Namen kombinieren, um einen neuen Themennamen zu erstellen.

In einem MQI-Programm wird der vollständige Themenname von MQOPEN erstellt. Er besteht aus zwei Feldern, die in Publish/Subscribe-MQI-Aufrufen verwendet werden, in der aufgeführten Reihenfolge:

1. Das Attribut **TOPICSTR** des Themenobjekts, das im Feld **ObjectName** benannt ist.
2. Der Parameter **ObjectString**, der das von der Anwendung bereitgestellte Unterthema definiert.

Die daraus resultierende Themenzeichenfolge wird im Parameter **ResObjectString** zurückgegeben.

Diese Felder gelten als vorhanden, wenn das erste Zeichen jedes Felds kein Leerzeichen oder Nullzeichen ist und die Feldlänge größer als null ist. Wenn nur eines der Felder vorhanden ist, wird es unverändert als Themenname verwendet. Wenn kein Feld einen Wert aufweist, schlägt der Aufruf mit dem Ursachencode MQRC_UNKNOWN_OBJECT_NAME oder MQRC_TOPIC_STRING_ERROR fehl, wenn der vollständige Themenname nicht gültig ist.

Wenn beide Felder vorhanden sind, wird ein '/'-Zeichen zwischen den beiden Elementen des entstehenden kombinierten Themennamens eingefügt.

Tabelle 548 auf Seite 569 zeigt Beispiele für die Verkettung von Themenzeichenfolgen:

Tabelle 548. Beispiele für die Verkettung von Themenzeichenfolgen

TOPICSTR	ObjectString	Vollständiger Themenname	Kommentar
Fußball/Ergebnisse	' '	Fußball/Ergebnisse	Die TOPICSTR wird alleine verwendet

Tabelle 548. Beispiele für die Verkettung von Themenzeichenfolgen (Forts.)			
TOPICSTR	**ObjectString**	**Vollständiger Themenname**	**Kommentar**
' '	Fußball/Ergebnisse	Fußball/Ergebnisse	Die ObjectString wird alleine verwendet
Fußball	Ergebnisse	Fußball/Ergebnisse	Ein '/'-Zeichen wird am Verkettungspunkt hinzugefügt
Fußball	/Ergebnisse	Fußball//Ergebnisse	Ein 'leerer Knoten' wird zwischen den beiden Zeichenfolgen erstellt
/Fußball	Ergebnisse	/Fußball/Ergebnisse	Das Thema beginnt mit einem 'leeren Knoten'

Das Zeichen '/' wird als Sonderzeichen betrachtet, das eine Struktur für den vollständigen Themennamen in Themenstrukturen bereitstellt, und darf aus anderen Gründen nicht verwendet werden, da die Struktur der Themenstruktur betroffen ist. Das Thema "`/Football`" entspricht nicht dem Thema "`Football`".

Die folgenden Platzhalterzeichen sind Sonderzeichen:

- Pluszeichen '+'
- Nummernzeichen '#'
- Stern '*'
- Fragezeichen '?'

Diese Zeichen werden nicht als ungültig betrachtet, Sie müssen sich jedoch mit ihrer Verwendung vertraut machen. Sie können darauf verzichten, diese Zeichen beim Veröffentlichen in Ihre Themenzeichenfolgen aufzunehmen. Das Veröffentlichen einer Themenzeichenfolge mit "#" oder "+" in Kombination mit anderen Zeichen (einschließlich ihrer selbst) innerhalb einer Themenebene kann durch beide Platzhalterschemata subskribiert werden. Die Veröffentlichung in einer Themenzeichenfolge mit '#' oder '+' als die einzigen Zeichen zwischen zwei '/'-Zeichen erzeugt eine Themenzeichenfolge, die von einer Anwendung, die das Platzhalterschema MQSO_WILDCARD_TOPIC verwendet, nicht explizit abonniert werden kann. Diese Situation führt dazu, dass die Anwendung mehr Veröffentlichungen abruft als erwartet.

Mustercodeausschnitt

Dieser Codeausschnitt, der aus dem Beispielprogramm Beispiel 2: Publisher eines variablen Themas stammt, kombiniert ein Themenobjekt mit einer variablen Themenzeichenfolge.

```
MQOD     td = {MQOD_DEFAULT};  /* Object Descriptor            */
td.ObjectType = MQOT_TOPIC;    /* Object is a topic            */
td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4    */
strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
td.ObjectString.VSPtr = topicString;
td.ObjectString.VSLength = (MQLONG)strlen(topicString);
td.ResObjectString.VSPtr = resTopicStr;
td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
```

Anfangswerte und Sprachendeklarationen für den Nachrichtendeskriptor

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQSD_STRUC_ID	'SD¬¬'
Version	MQSD_VERSION_1	1
Options	MQSO_NON_DURABLE	0

Name des Felds	Name der Konstante	Wert der Konstanten
ObjectName	--	Nullzeichenfolge oder Leerzeichen.
AlternateUserId	--	Nullzeichenfolge oder Leerzeichen.
AlternateSecurityId	MQSID_NONE	Nullen
SubExpiry	MQEI_UNLIMITED	-1
ObjectString	--	Namen und Werte gemäß der Definition für MQCHARV
SubName	--	Namen und Werte gemäß der Definition für MQCHARV
SubUserData	--	Namen und Werte gemäß der Definition für MQCHARV
SubCorrelId	MQCI_NONE	Nullen
PubPriority	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken	MQACT_NONE	Nullen
PubApplIdentityData	--	Nullzeichenfolge oder Leerzeichen.
Selection String	--	Namen und Werte gemäß der Definition für MQCHARV
SubLevel	--	1
ResObjectString	--	Namen und Werte gemäß der Definition für MQCHARV

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQSD_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQSD MySD = {MQSD_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options associated with subscribing */
  MQCHAR48  ObjectName;           /* Object name */
  MQCHAR12  AlternateUserId;      /* Alternate user identifier */
  MQBYTE40  AlternateSecurityId;  /* Alternate security identifier */
  MQLONG    SubExpiry;            /* Expiry of Subscription */
  MQCHARV   ObjectString;         /* Object Long name */
  MQCHARV   SubName;              /* Subscription name */
  MQCHARV   SubUserData;          /* Subscription User data */
  MQBYTE24  SubCorrelId;          /* Correlation Id related to this subscription */
  MQLONG    PubPriority;          /* Priority set in publications */
  MQBYTE32  PubAccountingToken;   /* Accounting Token set in publications */
  MQCHAR32  PubApplIdentityData;  /* Appl Identity Data set in publications */
```

```
  MQCHARV   SelectionString;      /* Message selector structure */
  MQLONG    SubLevel;             /* Subscription level */
  MQCHARV   ResObjectString;      /* Resolved Long object name*/
  /* Ver:1 */
};
```

COBOL-Delaration

```
** 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.
```

Deklaration in PL/I

```
dcl
 1 MQSD based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version number */
  3 Options             fixed bin(31), /* Options associated with subscribing */
  3 ObjectName          char(48),      /* Object name */
  3 AlternateUserId     char(12),      /* Alternate user identifier */
  3 AlternateSecurityId char(40),      /* Alternate security identifier */
  3 SubExpiry           fixed bin(31), /* Expiry of Subscription */
```

```
  3 ObjectString,                      /* Object Long name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31); /* CCSID of variable length string */
  3 SubName,                           /* Subscription name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31); /* CCSID of variable length string */
  3 SubUserData,                       /* Subscription User data */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 SubCorrelId         char(24),      /* Correlation Id related to this subscription */
  3 PubPriority         fixed bin(31), /* Priority set in publications */
  3 PubAccountingToken  char(32),      /* Accounting Token set in publications */
  3 PubApplIdentityData char(32),      /* Appl Identity Data set in publications */
  3 SelectionString,                   /* Message Selection */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 SubLevel            fixed bin(31), /* Subscription level */
  3 ResObjectString,                   /* Resolved Long object name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31); /* CCSID of variable length string */
```

Deklaration in High Level Assembler

```
MQSD                           DSECT
MQSD_STRUCID                   DS    CL4   Structure identifier
MQSD_VERSION                   DS    F     Structure version number
MQSD-OPTIONS                   DS    F     Options associated with subscribing
MQSD_OBJECTNAME                DS    CL48  Object name
MQSD_ALTERNATEUSERID           DS    CL12  Alternate user identifier
MQSD_ALTERNATESECURITYID       DS    CL40  Alternate security identifier
MQSD_SUBEXPIRY                 DS    F     Expiry of Subscription
MQSD_OBJECTSTRING              DS    0F    Object Long name
MQSD_OBJECTSTRING_VSPTR        DS    F     Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET     DS    F     Offset of variable length string
MQSD_OBJECTSTRING_VSBUFSIZE    DS    F     size of buffer
MQSD_OBJECTSTRING_VSLENGTH     DS    F     Length of variable length string
MQSD_OBJECTSTRING_VSCCSID      DS    F     CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH       EQU   *-MQSD_OBJECTSTRING
                               ORG   MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA         DS    CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME                   DS    0F    Subscription name
MQSD_SUBNAME_VSPTR             DS    F     Address of variable length string
MQSD_SUBNAME_VSOFFSET          DS    F     Offset of variable length string
MQSD_SUBNAME_VSBUFSIZE         DS    F     size of buffer
MQSD_SUBNAME_VSLENGTH          DS    F     Length of variable length string
MQSD_SUBNAME_VSCCSID           DS    F     CCSID of variable length string
MQSD_SUBNAME_LENGTH            EQU   *-MQSD_SUBNAME
                               ORG   MQSD_SUBNAME
MQSD_SUBNAME_AREA              DS    CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA               DS    0F    Subscription User data
MQSD_SUBUSERDATA_VSPTR         DS    F     Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET      DS    F     Offset of variable length string
MQSD_SUBUSERDATA_VSBUFSIZE     DS    F     size of buffer
MQSD_SUBUSERDATA_VSLENGTH      DS    F     Length of variable length string
MQSD_SUBUSERDATA_VSCCSID       DS    F     CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH        EQU   *-MQSD_SUBUSERDATA
                               ORG   MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA          DS    CL(MQSD_SUBUSERDATA_LENGTH)
```

```
*
MQSD_SUBCORRELID               DS    CL24  Correlation Id related to this subscription
MQSD_PUBPRIORITY               DS    F     Priority set in publications
MQSD_PUBACCOUNTINGTOKEN        DS    CL32  Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA       DS    CL32  Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING           DS   F     Message Selector
MQSD_SELECTIONSTRING_VSPTR     DS   F     Address of variable length string
MQSD_SELECTIONSTRING_VSOFFSET  DS   F     Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFSIZE DS   F     size of buffer
MQSD_SELECTIONSTRING_VSLENGTH  DS   F     Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID   DS   F     CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH    EQU  *- MQSD_SELECTIONSTRING
                               ORG  MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA      DS   CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL                  DS   F     Subscription level
*
MQSD_RESOBJECTSTRING           DS   F     Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR     DS   F     Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET  DS   F     Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFSIZE DS   F     size of buffer
MQSD_RESOBJECTSTRING_VSLENGTH  DS   F     Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID   DS   F     CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH    EQU  *- MQSD_RESOBJECTSTRING
                               ORG  MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA      DS   CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH                    EQU   *-MQSD
                               ORG   MQSD
MQSD_AREA                      DS    CL(MQSD_LENGTH)
```

MQSMPO – Festlegen von Optionen für Nachrichteneigenschaften

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 549. Felder in MQSMPO

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options
ValueEncoding	Eigenschaftswert Codierung	ValueEncoding
ValueCCSID	Eigenschaftswert Zeichensatz	ValueCCSID

Überblick für MQSMPO

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Mit der **MQSMPO**-Struktur können Anwendungen Optionen festlegen, die steuern, wie die Eigenschaften von Nachrichten festgelegt werden. Die Struktur ist ein Eingabeparameter für den **MQSETMP**-Aufruf.

Zeichensatz und Codierung: Die Daten in **MQSMPO** müssen im Zeichensatz und in der Codierung der Anwendung (**MQENC_NATIVE**) enthalten sein.

Felder für MQSMPO

Die MQSMPO-Struktur enthält die folgenden Felder (die Felder werden in **alphabetischer Reihenfolge** beschrieben):

Optionen (MQLONG)

Positionsoptionen: Die folgenden Optionen beziehen sich auf die relative Position der Eigenschaft verglichen mit dem Eigenschaftscursor:

MQSMPO_SET_FIRST
Legt den Wert der ersten mit dem angegebenen Namen übereinstimmenden Eigenschaft fest oder, falls sie nicht existiert, fügt eine neue Eigenschaft hinter allen anderen Eigenschaften mit einer übereinstimmenden Hierarchie hinzu.

MQSMPO_SET_PROP_UNDER_CURSOR
Legt den Wert der Eigenschaft fest, auf die der Eigenschaftscursor zeigt. Der Eigenschaftscursor verweist auf die letzte Eigenschaft, die über die Option MQIMPO_INQ_FIRST oder MQIMPO_INQ_NEXT abgefragt wurde.

Der Eigenschaftscursor wird zurückgesetzt, wenn die Nachrichtenkennung bei einem MQGET-Aufruf wiederverwendet wird oder wenn die Nachrichtenkennung im Feld *MsgHandle* der MQGMO- oder MQPMO-Struktur eines MQPUT-Aufrufs angegeben wird.

Wird diese Option verwendet, wenn noch kein Eigenschaftscursor eingerichtet wurde oder wenn die Eigenschaft, auf die der Eigenschaftscursor verweist, gelöscht wurde, schlägt der Aufruf mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl.

MQSMPO_SET_PROP_BEFORE_CURSOR
Legt eine neue Eigenschaft vor der Eigenschaft fest, auf die der Eigenschaftscursor verweist. Der Eigenschaftscursor verweist auf die letzte Eigenschaft, die über die Option MQIMPO_INQ_FIRST oder MQIMPO_INQ_NEXT abgefragt wurde.

Der Eigenschaftscursor wird zurückgesetzt, wenn die Nachrichtenkennung bei einem MQGET-Aufruf wiederverwendet wird oder wenn die Nachrichtenkennung im Feld *MsgHandle* der MQGMO- oder MQPMO-Struktur eines MQPUT-Aufrufs angegeben wird.

Wird diese Option verwendet, wenn noch kein Eigenschaftscursor eingerichtet wurde oder wenn die Eigenschaft, auf die der Eigenschaftscursor verweist, gelöscht wurde, schlägt der Aufruf mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl.

MQSMPO_SET_PROP_AFTER_CURSOR
Legt eine neue Eigenschaft hinter der Eigenschaft fest, auf die der Eigenschaftscursor zeigt. Der Eigenschaftscursor verweist auf die letzte Eigenschaft, die über die Option MQIMPO_INQ_FIRST oder MQIMPO_INQ_NEXT abgefragt wurde.

Der Eigenschaftscursor wird zurückgesetzt, wenn die Nachrichtenkennung bei einem MQGET-Aufruf wiederverwendet wird oder wenn die Nachrichtenkennung im Feld *MsgHandle* der MQGMO- oder MQPMO-Struktur eines MQPUT-Aufrufs angegeben wird.

Wird diese Option verwendet, wenn noch kein Eigenschaftscursor eingerichtet wurde oder wenn die Eigenschaft, auf die der Eigenschaftscursor verweist, gelöscht wurde, schlägt der Aufruf mit dem Beendigungscode MQCC_FAILED und dem Grund MQRC_PROPERTY_NOT_AVAILABLE fehl.

MQSMPO_APPEND_PROPERTY
Eine neue Eigenschaft wird nach allen anderen Eigenschaften mit einer übereinstimmenden Hierarchie hinzugefügt. Wenn mindestens eine Eigenschaft vorhanden ist, die mit dem angegebenen Namen übereinstimmt, wird am Ende dieser Liste mit Eigenschaften eine neue Eigenschaft hinzugefügt.

Durch diese Option kann eine Liste von Eigenschaften mit dem gleichen Namen erstellt werden.

Wenn Sie keine der beschriebenen Optionen benötigen, verwenden Sie folgende Option:

MQSMPO_NONE
Keine Optionen angegeben.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSMPO_SET_FIRST.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQSMPO_STRUC_ID

ID der Struktur zur Festlegung der Nachrichteneigenschaften.

Für die Programmiersprache C wird auch die Konstante **MQSMPO_STRUC_ID_ARRAY** definiert. Diese hat denselben Wert wie **MQSMPO_STRUC_ID**, ist aber eine Gruppe von Zeichen und keine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQSMPO_STRUC_ID**.

ValueCCSID (MQLONG)

Der Zeichensatz des Eigenschaftswerts, der festzulegen ist, wenn es sich bei dem Wert um eine Zeichenfolge handelt.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQCCSI_APPL**.

ValueEncoding (MQLONG)

Der Zeichensatz des Eigenschaftswerts, der festzulegen ist, wenn der Wert numerisch ist.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQENC_NATIVE**.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQSMPO_VERSION_1
Version-1 der Optionsstruktur zum Festlegen der Nachrichteneigenschaften.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQSMPO_CURRENT_VERSION
Aktuelle Version der Optionsstruktur zum Festlegen der Nachrichteneigenschaften.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist **MQSMPO_VERSION_1**.

Anfangswerte und Sprachendeklarationen für MQSMPO

Tabelle 550. Anfangswerte der Felder in MQSMPO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQSMPO_STRUC_ID	'SMPO'
Version	MQSMPO_VERSION_1	1
Options	MQSMPO_NONE	0
ValueEncoding	MQENC_NATIVE	Von der Umgebung abhängig
ValueCCSID	MQCCSI_APPL	-3

Tabelle 550. Anfangswerte der Felder in MQSMPO (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
2. In der Programmiersprache C enthält die MakrovariableMQSMPO_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQSETMP */
  MQLONG    ValueEncoding;   /* Encoding of Value */
  MQLONG    ValueCCSID;      /* Character set identifier of Value */
};
```

COBOL-Delaration

```
**   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.
```

Deklaration in PL/I

```
dcl
 1 MQSMPO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQSETMP */
  3 ValueEncoding  fixed bin(31), /* Encoding of Value */
  3 ValueCCSID     fixed bin(31), /* Character set identifier of Value */
```

Deklaration in High Level Assembler

```
MQSMPO                DSECT
MQSMPO_STRUCID        DS   CL4   Structure identifier
MQSMPO_VERSION        DS   F     Structure version number
MQSMPO_OPTIONS        DS   F     Options that control the action of
*                                MQSETMP
MQSMPO_VALUEENCODING  DS   F     Encoding of VALUE
MQSMPO_VALUECCSID     DS   F     Character set identifier of VALUE
MQSMPO_LENGTH         EQU  *-MQSMPO
MQSMPO_AREA           DS   CL(MQSMPO_LENGTH)
```

MQSRO - Optionen Subskriptionsanforderung

In diesem Abschnitt werden Optionen zur Subskriptionsanforderung, die enthaltenen Felder und die Anfangswerte dieser Felder beschrieben.

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
Options	Optionen	Options
NumPubs	Anzahl der Veröffentlichungen	NumPubs

Überblick für MQSRO

Verfügbarkeit: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS sowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

Zweck: Mit der MQSRO-Struktur kann die Anwendung Optionen angeben, die festlegen, wie eine Subskriptionsanforderung ausgeführt wird. Die Struktur ist ein Ein-/Ausgabeparameter im Aufruf MQSUBRQ.

Version: Die aktuelle Version von MQSRO ist MQSRO_VERSION_1.

Zeichensatz und Codierung: Daten in MQSRO müssen im Zeichensatz vorliegen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird, sowie in der Codierung des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angegeben wird. Wird die Anwendung allerdings als MQ MQI-Client ausgeführt, muss die Struktur im Zeichensatz und der Codierung des Clients enthalten sein.

Felder für MQSRO

Die MQSRO-Struktur enthält die folgenden Felder, die in alphabetischer Reihenfolge beschrieben werden:

NumPubs (MQLONG)

Dies ist ein Ausgabefeld, das zur Anwendung zurückgegeben wird, um die Anzahl der Veröffentlichungen anzugeben, die durch diesen Aufruf zur Subskriptionswarteschlange gesendet wurden. Auch wenn diese Anzahl an Veröffentlichungen als Ergebnis dieses Aufrufs gesendet wurde, gibt es keine Garantie dafür, dass so viele Nachrichten zum Abrufen durch die Anwendung verfügbar sind, insbesondere, wenn es sich um nicht persistente Nachrichten handelt.

Unter Umständen sind mehrere Veröffentlichungen verfügbar, wenn das abonnierte Thema ein Platzhalterzeichen enthält. Befanden sich in der Themenzeichenfolge keine Platzhalter, als die durch *HSUB* dargestellte Subskription erstellt wurde, wird durch diesen Aufruf höchstens eine Veröffentlichung gesendet.

Optionen (MQLONG)

Eine der folgenden Optionen muss angegeben werden. Es kann nur eine Option angegeben werden.

MQSRO_FAIL_IF_QUIESCING

Der Aufruf MQSUBRQ schlägt fehl, wenn sich der Warteschlangenmanager im Quiescestatus befindet. Unter z/OS erzwingt diese Option für eine CICS- oder IMS-Anwendung auch, dass der Aufruf MQSUBRQ fehlschlägt, wenn sich die Verbindung im Quiescestatus befindet.

Standardoption: Wenn die oben beschriebene Option nicht erforderlich ist, muss die folgende Option verwendet werden:

MQSRO_NONE

Dieser Wert bedeutet, dass keine anderen Optionen angegeben wurden; alle Optionen nehmen ihre Standardwerte an.

MQSRO_NONE unterstützt die Programmdokumentation. Diese Option ist zwar nicht zur Verwendung mit einer anderen Option gedacht; da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

StrucId (MQCHAR4)

Dies ist die Struktur-ID, die folgenden Wert haben muss:

MQSRO_STRUC_ID
ID für die Struktur Optionen Subskriptionsanforderung.

Für die Programmiersprache C ist auch die Konstante MQSRO_STRUC_ID_ARRAY definiert; diese Konstante hat denselben Wert wie MQSRO_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSRO_STRUC_ID.

Version (MQLONG)

Dies ist die Strukturversionsnummer, die folgenden Wert haben muss:

MQSRO_VERSION_1
Struktur Optionen Subskriptionsanforderung der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQSRO_CURRENT_VERSION
Aktuelle Version der Struktur Optionen Subskriptionsanforderung.

Dies ist immer ein Eingabefeld. Der Anfangswert dieses Felds ist MQSRO_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQSRO

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQSRO_STRUC_ID	'SRO¬'
Version	MQSRO_VERSION_1	1
Options	MQSRO_NONE	0
NumPubs	--	0

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQSRO_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options that control the action of MQSUBRQ */
  MQLONG    NumPubs;              /* Number of publications sent */
  /* Ver:1 */
};
```

COBOL-Delaration

```
** 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.
```

Deklaration in PL/I

```
dcl
 1 MQSRO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action of MQSUBRQ */
   3 NumPubs        fixed bin(31); /* Number of publications sent */
```

Deklaration in High Level Assembler

```
MQSRO                   DSECT
MQSRO_STRUCID           DS    CL4   Structure identifier
MQSRO_VERSION           DS    F     Structure version number
MQSRO_OPTIONS           DS    F     Options that control the action of MQSUBRQ
MQSRO_NUMPUBS           DS    F     Number of publications sent
*
MQSRO_LENGTH            EQU   *-MQSRO
                        ORG   MQSRO
MQSRO_AREA              DS    CL(MQSRO_LENGTH)
```

MQSTS – Statusberichtsstruktur

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 551. Felder für MQSTS

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
CompCode	Beendigungscode des ersten Fehlers	CompCode
Reason	Ursachencode des ersten Fehlers	Reason
PutSuccessCount	Anzahl erfolgreiche asynchrone Put-Aufrufe	SuccessCount
PutWarningCount	Anzahl asynchrone Put-Aufrufe mit Warnungen	WarningCount
PutFailureCount	Anzahl fehlgeschlagene asynchrone Put-Aufrufe	FailureCount
ObjectType	Typ des fehlgeschlagenen Objekts	ObjectType
ObjectName	Name des fehlgeschlagenen Objekts	ObjectName
ObjectQMgrName	Name des Warteschlangenmanagers, der Eigner des fehlgeschlagenen Objekts ist	ObjectQMgrName
ResolvedObjectName	Aufgelöster Name der Zielwarteschlange	ResolvedObjectName
ResolvedQMgrName	Aufgelöster Name des Zielwarteschlangenmanagers	ResolvedQMgrName

Tabelle 551. Felder für MQSTS (Forts.)		
Feld	**Beschreibung**	**Thema**
Anmerkung: Die übrigen Felder werden ignoriert, wenn Version kleiner als MQSTS_VERSION_2 ist.		
ObjectString	Langer Objektname des fehlgeschlagenen Objekts	ObjectString
SubName	Subskriptionsname der fehlgeschlagenen Subskription	SubName
OpenOptions	Öffnungsoptionen, die dem Fehler zugeordnet sind	OpenOptions
SubOptions	Subskriptionsoptionen, die dem Fehler zugeordnet sind	SubOptions

Überblick zu MQSTS

Zweck: Die MQSTS-Struktur ist ein Ausgabeparameter des MQSTAT-Befehls.

Zeichensatz und Codierung: Für Zeichendaten im MQSTS gilt der Zeichensatz des lokalen Warteschlangenmanagers, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist. Numerische Daten in MQSTS entsprechen der nativen Systemcodierung, die durch *Encoding* angegeben wird.

Verwendung: Mit dem MQSTAT-Befehl werden die Statusinformationen abgerufen. Diese Informationen werden in einer MQSTS-Struktur zurückgegeben. Informationen zu MQSTAT finden Sie in „MQSTAT - Statusinformationen abrufen" auf Seite 782.

Felder für MQSTS

Die MQSTS-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CompCode (MQLONG)
Der Beendigungscode der Operation, die zurückgemeldet wird.

Die Interpretation von CompCode ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Dies ist der Beendigungscode einer vorherigen asynchronen Put-Operation für das Objekt, das in ObjectName angegeben ist.

MQSTAT_TYPE_RECONNECTION

Wenn die Verbindung wiederhergestellt wird oder die Verbindungswiederholung fehlgeschlagen ist, ist dies der Beendigungscode, der die Verbindungswiederholung ausgelöst hat.

Wenn die Verbindung derzeit hergestellt ist, lautet der Wert MQCC_OK.

MQSTAT_TYPE_RECONNECTION_ERROR

Wenn die Wiederherstellung der Verbindung fehlgeschlagen ist, ist dies der Beendigungscode, der das Fehlschlagen der Verbindungswiederholung ausgelöst hat.

Wenn die Verbindung derzeit hergestellt ist oder wiederhergestellt wird, lautet der Wert MQCC_OK.

CompCode ist immer ein Ausgabefeld. Der Anfangswert ist MQCC_OK.

ObjectName (MQCHAR48)
Der Name des Objekts, das zurückgemeldet wird.

Die Interpretation von ObjectName ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Dies ist der Name der Warteschlange oder des Themas, das in der Put-Operation verwendet wird, deren Fehlschlagen in den Feldern *CompCode* und *Reason* in der MQSTS-Struktur gemeldet wird.

MQSTAT_TYPE_RECONNECTION

Wenn die Verbindung wiederhergestellt wird, ist dies der Name des Warteschlangenmanagers, der der Verbindung zugeordnet ist.

MQSTAT_TYPE_RECONNECTION_ERROR

Wenn die Wiederherstellung der Verbindung fehlgeschlagen ist, ist dies der Name des Objekts, das das Fehlschlagen der Verbindungswiederholung ausgelöst hat. Die Ursache für das Fehlschlagen wird in den Feldern *CompCode* und *Reason* in der MQSTS-Struktur gemeldet.

ObjectName ist ein Ausgabefeld. Der Anfangswert ist die Nullzeichenfolge in C und 48 Leerzeichen in anderen Programmiersprachen.

ObjectQMgrName (MQCHAR48)
Der Name des Warteschlangenmanagers, der zurückgemeldet wird.

Die Interpretation von ObjectQMgrName ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Gibt den Namen des Warteschlangenmanagers an, auf dem das Objekt *ObjectName* definiert ist. Ein Name, der bis zum ersten Nullzeichen oder dem Ende des Felds leer ist, gibt den Warteschlangenmanager an, mit dem die Anwendung verbunden ist, also den lokalen Warteschlangenmanager.

MQSTAT_TYPE_RECONNECTION

Leer.

MQSTAT_TYPE_RECONNECTION_ERROR

Wenn die Wiederherstellung der Verbindung fehlgeschlagen ist, ist dies der Name des Objekts, das das Fehlschlagen der Verbindungswiederholung ausgelöst hat. Die Ursache für das Fehlschlagen wird in den Feldern *CompCode* und *Reason* in der MQSTS-Struktur gemeldet.

ObjectQMgrName ist ein Ausgabefeld. Sein Wert ist die Nullzeichenfolge in C und 48 Leerzeichen in anderen Programmiersprachen.

ObjectString (MQCHARV)
Langer Objektname des Objekts, bei dem der Fehler auftrat und das Gegenstand des Berichts ist. Nur vorhanden in MQSTS Version 2 oder höher.

Die Interpretation von ObjectString ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Dies ist der lange Objektname der Warteschlange oder des Themas, die bzw. das bei der fehlgeschlagenen MQPUT-Operation verwendet wurde.

MQSTAT_TYPE_RECONNECTION

Zeichenfolge mit Nulllänge.

MQSTAT_TYPE_RECONNECTION_ERROR

Dies ist der lange Objektname des Objekts, welches das Fehlschlagen der Verbindungswiederholung verursachte.

ObjectString ist ein Ausgabefeld. Der Anfangswert ist eine Zeichenfolge mit Nulllänge.

ObjectType (MQLONG)
Der Typ des in *ObjectName* genannten Objekts, das zurückgemeldet wird.

Mögliche Werte von ObjectType sind unter „MQOT_* (Objekttypen und erweiterte Objekttypen)” auf Seite 147 aufgelistet.

ObjectType ist ein Ausgabefeld. Der Anfangswert ist MQOT_Q.

OpenOptions (MQLONG)
Die OpenOptions , die zum Öffnen des Objekts verwendet wird, über das berichtet wird. Nur vorhanden in MQSTS Version 2 oder höher.

Der Wert von OpenOptions ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Null

MQSTAT_TYPE_RECONNECTION

Null

MQSTAT_TYPE_RECONNECTION_ERROR

Verwendete OpenOptions, als das Fehlschlagen aufgetreten ist. Die Ursache für das Fehlschlagen wird in den Feldern *CompCode* und *Reason* in der MQSTS-Struktur gemeldet.

OpenOptions ist ein Ausgabefeld. Der Anfangswert ist null.

PutFailureCount (MQLONG)
Dies ist die Anzahl fehlgeschlagener asynchroner Put-Operationen.

Der Wert von PutFailureCount ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Die Anzahl der asynchronen Put-Operationen für das Objekt, das in der MQSTS-Struktur angegeben wird, die mit MQCC_FAILED beendet wurden.

MQSTAT_TYPE_RECONNECTION

Null

MQSTAT_TYPE_RECONNECTION_ERROR

Null

PutFailureCount ist ein Ausgabefeld. Der Anfangswert ist null.

PutSuccessCount (MQLONG)
Dies ist die Anzahl erfolgreicher asynchroner Put-Operationen.

Der Wert von PutSuccessCount ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Die Anzahl der asynchronen Put-Operationen für das Objekt, das in der MQSTS-Struktur angegeben wird, die mit MQCC_OK beendet wurden.

MQSTAT_TYPE_RECONNECTION

Null

MQSTAT_TYPE_RECONNECTION_ERROR

Null

PutSuccessCount ist ein Ausgabefeld. Der Anfangswert ist null.

PutWarningCount (MQLONG)
Die Anzahl der asynchronen Put-Operationen, die mit einer Warnung beendet wurden.

Der Wert von PutWarningCount ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Die Anzahl der asynchronen Put-Operationen für das Objekt, das in der MQSTS-Struktur angegeben wird, die mit MQCC_WARNING beendet wurden.

MQSTAT_TYPE_RECONNECTION

Null

MQSTAT_TYPE_RECONNECTION_ERROR

Null

PutWarningCount ist ein Ausgabefeld. Der Anfangswert ist null.

SubName (MQCHARV)
Dies ist der Name der fehlgeschlagenen Subskription. Nur vorhanden in MQSTS Version 2 oder höher.

Die Interpretation von SubName ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Zeichenfolge mit Nulllänge

MQSTAT_TYPE_RECONNECTION

Zeichenfolge mit Nulllänge

MQSTAT_TYPE_RECONNECTION_ERROR

Der Name der Subskription, die das Fehlschlagen der Verbindungswiederholung verursachte. Wenn kein Subskriptionsname vorhanden ist oder das Fehlschlagen nicht mit einer Subskription in Verbindung steht, ist dies eine Zeichenfolge mit Nulllänge.

SubName ist ein Ausgabefeld. Der Anfangswert ist eine Zeichenfolge mit Nulllänge.

SubOptions (MQLONG)
Der SubOptions , der zum Öffnen der fehlgeschlagenen Subskription verwendet wird Nur vorhanden in MQSTS Version 2 oder höher.

Die Interpretation von SubOptions ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Null

MQSTAT_TYPE_RECONNECTION

Null

MQSTAT_TYPE_RECONNECTION_ERROR

Verwendete SubOptions, als das Fehlschlagen aufgetreten ist. Wenn das Fehlschlagen nicht mit der Subskription eines Themas zusammenhängt, ist der zurückgegebene Wert null.

SubOptions ist ein Ausgabefeld. Der Anfangswert ist null.

Ursache (MQLONG)
Der Ursachencode der Operation, die zurückgemeldet wird.

Die Interpretation von Reason ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

Dies ist der Ursachencode einer vorherigen asynchronen Put-Operation für das Objekt, das in ObjectName angegeben ist.

MQSTAT_TYPE_RECONNECTION

Wenn die Verbindung wiederhergestellt wird oder die Verbindungswiederholung fehlgeschlagen ist, ist dies der Ursachencode, der die Verbindungswiederholung ausgelöst hat.

Wenn die Verbindung derzeit hergestellt ist, lautet der Wert MQRC_NONE.

MQSTAT_TYPE_RECONNECTION_ERROR

Wenn die Wiederherstellung der Verbindung fehlgeschlagen ist, ist dies der Ursachencode, der das Fehlschlagen der Verbindungswiederholung ausgelöst hat.

Wenn die Verbindung derzeit hergestellt ist oder wiederhergestellt wird, lautet der Wert MQRC_NONE.

Reason ist ein Ausgabefeld. Der Anfangswert ist MQRC_NONE.

ResolvedObjectName (MQCHAR48)
Der Name des Objekts, das in *ObjectName* angegeben wird, nachdem der lokale Warteschlangenmanager den Namen aufgelöst hat.

Die Interpretation von ResolvedObjectName ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

ResolvedObjectName ist der Name des Objekts, das in *ObjectName* angegeben wird, nachdem der lokale Warteschlangenmanager den Namen aufgelöst hat. Der zurückgegebene Name ist der Name eines Objekts in dem Warteschlangenmanager, der durch *ResolvedQMgrName* angegeben wird.

MQSTAT_TYPE_RECONNECTION

Leer.

MQSTAT_TYPE_RECONNECTION_ERROR

Leer.

ResolvedObjectName ist ein Ausgabefeld. Der Anfangswert ist die Nullzeichenfolge in C und 48 Leerzeichen in anderen Programmiersprachen.

ResolvedQMgrName (MQCHAR48)
Der Name des Ziel-Warteschlangenmanagers, nachdem der lokale Warteschlangenmanager den Namen aufgelöst hat.

Die Interpretation von ResolvedQMgrName ist abhängig vom Wert des Parameters MQSTAT Type.

MQSTAT_TYPE_ASYNC_ERROR

ResolvedQMgrName ist der Name des Ziel-Warteschlangenmanagers, nachdem der lokale Warteschlangenmanager den Namen aufgelöst hat. Der zurückgegebene Name ist der Name des Warteschlangenmanagers, der Eigner des Objekts ist, das durch *ResolvedObjectName* angegeben wird. *ResolvedQMgrName* kann der Name des lokalen Warteschlangenmanagers sein.

MQSTAT_TYPE_RECONNECTION

Leer.

MQSTAT_TYPE_RECONNECTION_ERROR

Leer.

ResolvedQMgrName ist immer ein Ausgabefeld. Der Anfangswert ist die Nullzeichenfolge in C und 48 Leerzeichen in anderen Programmiersprachen.

StrucId (MQCHAR4)
Die ID für die Statusberichtsstruktur MQSTS.

StrucId ist die Struktur-ID. Folgende Werte sind möglich:

MQSTS_STRUC_ID
ID für Statusberichtsstruktur.

Für die Programmiersprache C ist auch die Konstante MQSTS_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie MQSTS_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

StrucId ist immer ein Eingabefeld. Der Anfangswert ist MQSTS_STRUC_ID.

Version (MQLONG)
Die Versionsnummer der Struktur.

Der Wert muss wie folgt angegeben werden:

MQSTS_VERSION_1
Statusberichtsstruktur der Version 1.

MQSTS_VERSION_2
Statusberichtsstruktur der Version 2.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQSTS_CURRENT_VERSION
Aktuelle Version der Statusberichtsstruktur. Die aktuelle Version ist `MQSTS_VERSION_2`.

`Version` ist immer ein Eingabefeld. Der Anfangswert ist `MQSTS_VERSION_1`.

Anfangswerte und Sprachendeklarationen für MQSTS

Tabelle 552. Anfangswerte der Felder im MQSTS

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQSTS_STRUC_ID	'STAT¬'
Version	MQSTS_VERSION_1	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
PutSuccessCount	--	0
PutWarningCount	--	0
PutFailureCount	--	0
ObjectType	MQOT_Q	1
ObjectName	--	Nullzeichenfolge oder Leerzeichen.
ObjectQMgrName	--	Nullzeichenfolge oder Leerzeichen.
ResolvedObjectName	--	Nullzeichenfolge oder Leerzeichen.
ResolvedQMgrName	--	Nullzeichenfolge oder Leerzeichen.
ObjectString	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
SubName	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
OpenOptions	--	0
SubOptions	--	0

Tabelle 552. Anfangswerte der Felder im MQSTS (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die Makrovariable MQSTS_DEFAULT die oben aufgelisteten Werte. Sie kann folgendermaßen verwendet werden, um Anfangswerte für die Felder in der Struktur bereitzustellen:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
 MQCHAR4   StrucId;            /* Structure identifier */
 MQLONG    Version;            /* Structure version number */
 MQLONG    CompCode;           /* Completion Code of first error */
 MQLONG    Reason;             /* Reason Code of first error */
 MQLONG    PutSuccessCount;    /* Number of Async calls succeeded */
 MQLONG    PutWarningCount;    /* Number of Async calls had warnings */
 MQLONG    PutFailureCount;    /* Number of Async calls had failures */
 MQLONG    ObjectType;         /* Failing object type */
 MQCHAR48  ObjectName;         /* Failing object name */
 MQCHAR48  ObjectQMgrName;     /* Failing object queue manager name */
 MQCHAR48  ResolvedObjectName; /* Resolved name of destination queue */
 MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination qmgr */
/* Ver:1 */
 MQCHARV   ObjectString;       /* Failing object long name */
 MQCHARV   SubName;            /* Failing subscription name */
 MQLONG    OpenOptions;        /* Failing open options */
 MQLONG    SubOptions;         /* Failing subscription options */
/* Ver:2 */
};
```

COBOL-Delaration

```
      ** 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 **
```

Deklaration in PL/I

```
dcl
   1 MQSTS based,
     3 StrucId             char(4),       /* Structure identifier */
      3 Version            fixed bin(31), /* Structure version number */
      3 CompCode           fixed bin(31), /* Completion code */
      3 Reason             fixed bin(31), /* Reason code */
      3 PutSuccessCount    fixed bin(31), /* Put success count */
      3 PutWarningCount    fixed bin(31), /* Put warning count */
      3 PutFailureCount    fixed bin(31), /* Put failure count */
      3 ObjectType         fixed bin(31), /* Object type */
      3 ObjectName         char(48), /* Object name */
      3 ObjectQmgrName     char(48), /* Object queue manager */
      3 ResolvedObjectName char(48), /* Resolved Object name */
      3 ResolvedQmgrName   char(48); /* Resolved Object queue manager */
  /* Ver:1 */
      3 ObjectString,                /* Failing object long name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
      3 SubName,                     /* Failing subscription name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
     3 OpenOptions fixed bin(31),    /* Failing open options */
     3 SubOptions fixed bin(31);     /* Failing subscription options */
    /* Ver:2 */
```

Deklaration in High Level Assembler

```
     MQSTS                          DSECT
     MQSTS_STRUCID                  DS    CL4   Structure identifier
     MQSTS_VERSION                  DS    F     Structure version number
     MQSTS_COMPCODE                 DS    F     Completion code
     MQSTS_REASON                   DS    F     Reason code
     MQSTS_PUTSUCCESSCOUNT          DS    F     Success count
     MQSTS_PUTWARNINGCOUNT          DS    F     Warning count
     MQSTS_PUTFAILURECOUNT          DS    F     Failure count
     MQSTS_OBJTYPE                  DS    F     Object type
     MQSTS_OBJNAME                  DS    CL48  Object name
     MQSTS_OBJQMGR                  DS    CL48  Object queue manager
     MQSTS_ROBJNAME                 DS    CL48  Resolved object name
     MQSTS_ROBJQMGR                 DS    CL48  Resolved object queue manager
     MQSTS_OBJECTSTRING             DS    0F   Force fullword alignment
     MQSTS_OBJECTSTRING_VSPTR       DS    A  Address of variable length string
     MQSTS_OBJECTSTRING_VSOFFSET    DS    F  Offset of variable length string
```

```
    MQSTS_OBJECTSTRING_VSBUFSIZE  DS    F  Size of buffer
    MQSTS_OBJECTSTRING_VSLENGTH   DS    F  Length of variable length string
    MQSTS_OBJECTSTRING_VSCCSID    DS    F  CCSID of variable length string
    MQSTS_OBJECTSTRING_LENGTH     EQU   *-MQSTS_OBJECTSTRING
                                        ORG   MQSTS_OBJECTSTRING
    MQSTS_OBJECTSTRING_AREA       DS    CL(MQSTS_OBJECTSTRING_LENGTH)
    *
    MQSTS_SUBNAME                 DS    0F Force fullword alignment
    MQSTS_SUBNAME_VSPTR           DS    A  Address of variable length string
    MQSTS_SUBNAME_VSOFFSET        DS    F  Offset of variable length string
    MQSTS_SUBNAME_VSBUFSIZE       DS    F  Size of buffer
    MQSTS_SUBNAME_VSLENGTH        DS    F  Length of variable length string
    MQSTS_SUBNAME_VSCCSID         DS    F  CCSID of variable length string
    MQSTS_SUBNAME_LENGTH          EQ    *-MQSTS_SUBNAME
                                        ORG   MQSTS_SUBNAME
    MQSTS_SUBNAME_AREA            DS    CL(MQSTS_SUBNAME_LENGTH)
    *
    MQSTS_OPENOPTIONS             DS    F  Failing open options
    MQSTS_SUBOPTIONS              DS    F  Failing subscription option
    MQSTS_LENGTH                  EQU   *-MQSTS
                                  ORG   MQSTS
    MQSTS_AREA              DS    CL(MQSTS_LENGTH)
```

MQTM - Auslösenachricht

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 553. Felder für MQTM

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
QName	Name der ausgelösten Warteschlange	QName
ProcessName	Name des Prozessobjekts	ProcessName
TriggerData	Daten des Auslösers	TriggerData
ApplType	Anwendungstyp	ApplType
ApplId	Anwendungs-ID	ApplId
EnvData	Umgebungsdaten	EnvData
UserData	Benutzerdaten	UserData

Überblick zu MQTM

Zweck: Die MQTM-Struktur beschreibt die Daten in der Auslösenachricht, die vom Warteschlangenmanager an eine Trigger-Monitor-Anwendung gesendet werden, wenn ein Auslöserereignis für eine Warteschlange auftritt.

Diese Struktur ist Teil des WebSphere MQ Trigger Monitor Interface (TMI), das eine der WebSphere MQ-Framework-Schnittstellen ist.

Formatname: MQFMT_TRIGGER.

Zeichensatz und Codierung: Der Zeichensatz der Zeichendaten in MQTM entspricht dem des Warteschlangenmanagers, der die MQTM generiert. Numerische Daten in MQTM entsprechen der Systemcodierung des Warteschlangenmanagers, der die MQTM generiert.

Der Zeichensatz und die Codierung von MQTM werden durch die Felder *CodedCharSetId* und *Encoding* angegeben:

- Im MQMD (wenn sich die MQTM-Struktur am Anfang der Nachrichtendaten befindet) oder
- In der Header-Struktur, die der MQTM-Struktur vorangeht (alle anderen Fälle).

Verwendung: Die Auslösemonitoranwendung muss einige oder alle Informationen in der Auslösenachricht an die Anwendung übergeben, die die Auslösemonitoranwendung startet. Informationen, die von der gestarteten Anwendung benötigt werden, umfassen *QName*, *TriggerData* und *UserData*. Die Auslösemonitoranwendung kann die MQTM-Struktur direkt an die gestartete Anwendung übergeben oder stattdessen eine MQTMC2-Struktur übergeben - je nachdem, was durch die Umgebung gestattet ist und sich für die gestartete Anwendung am besten eignet. Informationen zu MQTMC2 finden Sie in „MQTMC2 - Auslösenachricht 2 (Zeichenformat)" auf Seite 596.

- Unter z/OS: Bei einer MQAT_CICS-Anwendung, die mit der CKTI-Transaktion gestartet wird, wird die gesamte Auslösenachrichtstruktur der gestarteten Transaktion verfügbar gemacht. Die Informationen können mit dem Befehl EXEC CICS RETRIEVE abgerufen werden.
- Bei IBM i übergibt die mit WebSphere MQ bereitgestellte Auslösemonitoranwendung eine MQTMC2-Struktur an die gestartete Anwendung.

Informationen zur Verwendung von Auslösern finden Sie unter WebSphere MQ -Anwendungen mit Auslösern starten .

MQMD für eine Auslösenachricht: Die Felder im MQMD einer durch den Warteschlangenmanager generierten Auslösenachricht werden wie folgt eingestellt:

Feld im MQMD	Verwendeter Wert
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	MQMT_DATAGRAM
Expiry	MQEI_UNLIMITED
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	Warteschlangenmanagerattribut *CodedCharSetId*
Format	MQFMT_TRIGGER
Priority	Attribut *DefPriority* der Initialisierungswarteschlange
Persistence	MQPER_NOT_PERSISTENT
MsgId	Ein eindeutiger Wert
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	Leerzeichen
ReplyToQMgr	Warteschlangenmanagername
UserIdentifier	Leerzeichen
AccountingToken	MQACT_NONE
ApplIdentityData	Leerzeichen
PutApplType	MQAT_QMGR oder wie es dem Nachrichtenkanalagenten entspricht
PutApplName	Erste 28 Byte des Warteschlangenmanagernamens
PutDate	Sendedatum der Auslösenachricht
PutTime	Sendeuhrzeit der Auslösenachricht
ApplOriginData	Leerzeichen

Eine Anwendung, die Auslösenachrichten generiert, sollte ähnliche Werte festlegen, jedoch mit folgenden Ausnahmen:

- Das Feld *Priority* kann auf MQPRI_PRIORITY_AS_Q_DEF festgelegt werden (der Warteschlangenmanager ändert dies auf die Standardpriorität für die Initialisierungswarteschlange, wenn die Nachricht eingereiht wird).
- Das Feld *ReplyToQMgr* kann auf Leerzeichen festgelegt werden (der Warteschlangenmanager ändert dies auf den Namen des lokalen Warteschlangenmanagers, wenn die Nachricht eingereiht wird).
- Legen Sie die Kontextfelder so fest, wie es für die Anwendung erforderlich ist.

Felder für MQTM

Die MQTM-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

ApplId (MQCHAR256)

Dies ist eine Zeichenfolge, welche die zu startende Anwendung angibt. Sie wird von der Auslösemonitoranwendung verwendet, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *ApplId* des Prozessobjekts, das durch das Feld *ProcessName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Prozessdefinitionen” auf Seite 869. Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager.

Der Inhalt von *ApplId* richtet sich nach der Auslösemonitoranwendung. Der von WebSphere MQ bereitgestellte Auslösemonitor verlangt, dass *ApplId* der Name eines ausführbaren Programms ist. Die folgenden Hinweise gelten für die jeweiligen Umgebungen:

- Unter z/OS ist *ApplId*:
  - Eine CICS-Transaktions-ID für Anwendungen, die mit der CICS-Auslösemonitortransaktion CKTI gestartet werden
  - Eine IMS-Transaktions-ID für Anwendungen, die mithilfe des IMS-Auslösemonitors CSQQTRMN gestartet werden
- Auf Windows-Systemen kann dem Programmnamen ein Laufwerk und ein Verzeichnispfad vorangestellt werden.
- Unter IBM i kann dem Programmnamen ein Bibliotheksname und das Zeichen / vorangestellt werden.
- Auf UNIX-Systemen kann dem Programmnamen ein Verzeichnispfad vorangestellt werden.

Die Länge dieses Felds wird durch MQ_PROCESS_APPL_ID_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 256 Leerzeichen in anderen Programmiersprachen.

ApplType (MQLONG)

Dieses Feld gibt die Art des zu startenden Programms an. Es wird von der Auslösemonitoranwendung verwendet, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *ApplType* des Prozessobjekts, das durch das Feld *ProcessName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Prozessdefinitionen” auf Seite 869. Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager.

ApplType kann einen der folgenden Standardwerte haben: Benutzerdefinierter Typen können ebenfalls verwendet werden, sollten aber auf Werte im Bereich von MQAT_USER_FIRST bis MQAT_USER_LAST beschränkt werden:

MQAT_AIX
AIX-Anwendung (selber Wert wie MQAT_UNIX)

MQAT_BATCH
Stapelanwendung

MQAT_BROKER
Brokeranwendung

MQAT_CICS
CICS-Transaktion

MQAT_CICS_BRIDGE
CICS-Brückenanwendung

MQAT_CICS_VSE
CICS/VSE-Transaktion

MQAT_DOS
WebSphere MQ MQI-Clientanwendung auf PC DOS

MQAT_IMS
IMS-Anwendung

MQAT_IMS_BRIDGE
IMS-Brückenanwendung

MQAT_JAVA
Java-Anwendung

MQAT_MVS
MVS- oder TSO-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_NOTES_AGENT
Lotus Notes Agent-Anwendung

MQAT_NSK
HP Integrity NonStop Server-Anwendung

MQAT_OS390
OS/390-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_OS400
IBM i-Anwendung

MQAT_RRS_BATCH
RRS-Stapelanwendung

MQAT_UNIX
UNIX-Anwendung

MQAT_UNKNOWN
Unbekannter Anwendungstyp

MQAT_USER
Benutzerdefinierter Anwendungstyp

MQAT_VOS
Stratus VOS-Anwendung.

MQAT_WINDOWS
16-Bit-Windows-Anwendung

MQAT_WINDOWS_NT
32-Bit-Windows-Anwendung

MQAT_WLM
z/OS-Workload-Manager-Anwendung

MQAT_XCF
XCF.

MQAT_ZOS
z/OS-Anwendung

MQAT_USER_FIRST
Niedrigster Wert für einen benutzerdefinierten Anwendungstyp

MQAT_USER_LAST
Höchster Wert für einen benutzerdefinierten Anwendungstyp

Der Anfangswert dieses Felds ist `0`.

EnvData (MQCHAR128)

Dies ist eine Zeichenfolge mit Umgebungsinformationen zur zu startenden Anwendung. Sie wird von der Auslösemonitoranwendung verwendet, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *EnvData* des Prozessobjekts, das durch das Feld *ProcessName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Prozessdefinitionen" auf Seite 869. Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager.

Unter z/OS werden diese Informationen bei einer CICS-Anwendung, die mit der Transaktion CKTI gestartet wurde, oder bei einer IMS-Anwendung, die mit der Transaktion CSQQTRMN gestartet wurde, nicht verwendet.

Die Länge dieses Felds wird durch MQ_PROCESS_ENV_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 128 Leerzeichen in anderen Programmiersprachen.

ProcessName (MQCHAR48)

Dies ist der Name des Warteschlangenmanagerprozessobjekts, das für die ausgelöste Warteschlange angegeben ist, und kann von der Auslösemonitoranwendung verwendet werden, die die Auslösenachricht empfängt. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *ProcessName* der Warteschlange, die durch das Feld *QName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Warteschlangen" auf Seite 833.

Namen, die kürzer als die festgelegte Länge des Felds sind, werden immer rechts mit Leerzeichen aufgefüllt. Sie werden nicht vorzeitig durch ein Nullzeichen beendet.

Die Länge dieses Felds wird durch MQ_PROCESS_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

QName (MQCHAR48)

Dies ist der Name der Warteschlange, für die ein Auslöseereignis auftrat. Er wird für die Auslösemonitoranwendung verwendet, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *QName* der ausgelösten Warteschlange. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Warteschlangen" auf Seite 833.

Namen, die kürzer als die festgelegte Länge des Felds sind, werden rechts mit Leerzeichen aufgefüllt. Sie werden nicht vorzeitig durch ein Nullzeichen beendet.

Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQTM_STRUC_ID
ID für die Auslösenachrichtstruktur

Für die Programmiersprache C ist auch die Konstante MQTM_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQTM_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQTM_STRUC_ID.

TriggerData (MQCHAR64)

Dies sind in einem freien Format gehaltene Daten für die Auslösemonitoranwendung, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *TriggerData* der Warteschlange, die durch das Feld *QName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Warteschlangen" auf Seite 833. Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager.

Unter z/OS werden diese Informationen für eine CICS-Anwendung, die mit der Transaktion CKTI gestartet wurde, nicht verwendet.

Die Länge dieses Felds wird durch MQ_TRIGGER_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 64 Leerzeichen in anderen Programmiersprachen.

UserData (MQCHAR128)

Dies ist eine Zeichenfolge mit für die zu startende Anwendung relevanten Benutzerdaten. Sie wird von der Auslösemonitoranwendung verwendet, welche die Auslösenachricht erhält. Der Warteschlangenmanager initialisiert dieses Feld mit dem Wert des Attributs *UserData* des Prozessobjekts, das durch das Feld *ProcessName* angegeben ist. Weitere Informationen zu diesem Attribut finden Sie unter „Attribute für Prozessdefinitionen" auf Seite 869. Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager.

Unter Microsoft Windows darf die Zeichenfolge keine Anführungszeichen enthalten, wenn die Prozessdefinition an **runmqtrm** übergeben wird.

Die Länge dieses Felds wird durch MQ_PROCESS_USER_DATA_LENGTH angegeben. Der Anfangswert dieses Felds ist die Nullzeichenfolge in C und 128 Leerzeichen in anderen Programmiersprachen.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQTM_VERSION_1
Versionsnummer der Auslösenachrichtenstruktur.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQTM_CURRENT_VERSION
Aktuelle Version der Auslösenachrichtenstruktur.

Der Anfangswert dieses Felds ist MQTM_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQTM

Tabelle 554. Anfangswerte für Felder in der MQTM

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQTM_STRUC_ID	'TM¬¬'
Version	MQTM_VERSION_1	1
QName	--	Nullzeichenfolge oder Leerzeichen.
ProcessName	--	Nullzeichenfolge oder Leerzeichen.
TriggerData	--	Nullzeichenfolge oder Leerzeichen.
ApplType	--	0
ApplId	--	Nullzeichenfolge oder Leerzeichen.
EnvData	--	Nullzeichenfolge oder Leerzeichen.
UserData	--	Nullzeichenfolge oder Leerzeichen.

Tabelle 554. Anfangswerte für Felder in der MQTM (Forts.)

Name des Felds	Name der Konstante	Wert der Konstanten

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQTM_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQLONG     Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQLONG     ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQTM based,
   3 StrucId     char(4),       /* Structure identifier */
   3 Version     fixed bin(31), /* Structure version number */
   3 QName       char(48),      /* Name of triggered queue */
   3 ProcessName char(48),      /* Name of process object */
   3 TriggerData char(64),      /* Trigger data */
   3 ApplType    fixed bin(31), /* Application type */
   3 ApplId      char(256),     /* Application identifier */
   3 EnvData     char(128),     /* Environment data */
   3 UserData    char(128);     /* User data */
```

Deklaration in High Level Assembler

```
MQTM              DSECT
MQTM_STRUCID      DS   CL4    Structure identifier
MQTM_VERSION      DS   F      Structure version number
MQTM_QNAME        DS   CL48   Name of triggered queue
MQTM_PROCESSNAME  DS   CL48   Name of process object
MQTM_TRIGGERDATA  DS   CL64   Trigger data
MQTM_APPLTYPE     DS   F      Application type
MQTM_APPLID       DS   CL256  Application identifier
MQTM_ENVDATA      DS   CL128  Environment data
MQTM_USERDATA     DS   CL128  User data
*
MQTM_LENGTH       EQU  *-MQTM
                  ORG  MQTM
MQTM_AREA         DS   CL(MQTM_LENGTH)
```

Deklaration in Visual Basic

```
Type MQTM
  StrucId     As String*4   'Structure identifier'
  Version     As Long       'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As Long       'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
End Type
```

MQTMC2 - Auslösenachricht 2 (Zeichenformat)

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 555. Felder für MQTMC2

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
QName	Name der ausgelösten Warteschlange	QName
ProcessName	Name des Prozessobjekts	ProcessName
TriggerData	Daten des Auslösers	TriggerData
ApplType	Anwendungstyp	ApplType
ApplId	Anwendungs-ID	ApplId
EnvData	Umgebungsdaten	EnvData
UserData	Benutzerdaten	UserData
QMgrName	Name des Warteschlangenmanagers	QMgrName

Überblick zu MQTMC2

Zweck: Wenn eine Auslösemonitor-Anwendung aus einer Initialisierungswarteschlange eine Auslösenachricht (MQTM) abruft, muss der Auslösemonitor möglicherweise einige oder alle Informationen in der Auslösenachricht an die Anwendung übergeben, die der Auslösemonitor startet.

Zu den Informationen, die die gestartete Anwendung möglicherweise benötigt, gehören *QName*, *TriggerData* und *UserData*. Die Auslösemonitoranwendung kann die MQTM-Struktur direkt an die gestar-

tete Anwendung übergeben oder stattdessen eine MQTMC2-Struktur, abhängig davon, was die Umgebung zulässt oder was der gestarteten Anwendung entspricht.

Diese Struktur ist Teil des WebSphere MQ Trigger Monitor Interface (TMI), das eine der WebSphere MQ-Framework-Schnittstellen ist.

Zeichensatz und Codierung: Der Zeichensatz der Zeichendaten in MQTMC2 entspricht dem des lokalen Warteschlangenmanagers, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird.

Verwendung: Die MQTMC2-Struktur ist dem Format der MQTM-Struktur sehr ähnlich. Der Unterschied besteht darin, dass die Nicht-Zeichenfelder in MQTM in MQTMC2 zu Zeichenfeldern derselben Länge geändert werden, und dass der Warteschlangenmanagername am Ende der Struktur hinzugefügt wird.

- Unter z/OS wird, wenn eine MQAT_IMS-Anwendungmit der CSQQTRMN-Anwendung gestartet wird, der gestarteten Anwendung eine MQTMC2-Struktur zur Verfügung gestellt.
- Unter IBM i: Die Auslösemonitoranwendung, die mit WebSphere MQ bereitgestellt wird, übergibt eine MQTMC2-Struktur an die gestartete Anwendung.

Felder für MQTMC2

Die MQTMC2-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

ApplId (MQCHAR256)

Anwendungskennung.

Siehe das Feld *ApplId* in der MQTM-Struktur.

ApplType (MQCHAR4)
Anwendungstyp.

Dieses Feld enthält immer Leerzeichen, unabhängig vom Wert im Feld *ApplType* in der MQTM-Struktur der ursprünglichen Auslösenachricht.

EnvData (MQCHAR128)
Umgebungsdaten.

Siehe das Feld *EnvData* in der MQTM-Struktur.

ProcessName (MQCHAR48)
Name des Prozessobjekts.

Siehe das Feld *ProcessName* in der MQTM-Struktur.

QMgrName (MQCHAR48)
Warteschlangenmanagername.

Dies ist der Name des Warteschlangenmanagers, wo das Auslöseereignis auftrat.

QName (MQCHAR48)
Name der ausgelösten Warteschlange.

Siehe das Feld *QName* in der MQTM-Struktur.

StrucId (MQCHAR4)
Struktur-ID.

Folgende Werte sind möglich:

MQTMC_STRUC_ID
ID für Auslösenachrichtstruktur (Zeichenformat)

Für die Programmiersprache C ist auch die Konstante MQTMC_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQTMC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

TriggerData (MQCHAR64)
Auslöserdaten.

Siehe das Feld *TriggerData* in der MQTM-Struktur.

UserData (MQCHAR128)
Benutzerdaten.

Siehe das Feld *UserData* in der MQTM-Struktur.

Version (MQCHAR4)
Strukturversionsnummer.

Folgende Werte sind möglich:

MQTMC_VERSION_2
Auslösenachrichtstruktur der Version 2 (Zeichenformat)

Für die Programmiersprache C ist auch die Konstante MQTMC_VERSION_2_ARRAY definiert. Sie hat den gleichen Wert wie die MQTMC_VERSION_2, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQTMC_CURRENT_VERSION
Aktuelle Version der Struktur der Auslösenachricht (Zeichenformat).

Anfangswerte und Sprachendeklarationen für MQTMC2

Tabelle 556. Anfangswerte für Felder in MQTMC2

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQTMC_STRUC_ID	'TMC¬'
Version	MQTMC_VERSION_2	'¬¬¬2'
QName	--	Nullzeichenfolge oder Leerzeichen.
ProcessName	--	Nullzeichenfolge oder Leerzeichen.
TriggerData	--	Nullzeichenfolge oder Leerzeichen.
ApplType	--	Leerzeichen
ApplId	--	Nullzeichenfolge oder Leerzeichen.
EnvData	--	Nullzeichenfolge oder Leerzeichen.
UserData	--	Nullzeichenfolge oder Leerzeichen.

Tabelle 556. Anfangswerte für Felder in MQTMC2 (Forts.)		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**
QMgrName	--	Nullzeichenfolge oder Leerzeichen.

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQTMC2_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQCHAR4    Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQCHAR4    ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
  MQCHAR48   QMgrName;     /* Queue manager name */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQTMC2 based,
  3 StrucId     char(4),   /* Structure identifier */
  3 Version     char(4),   /* Structure version number */
  3 QName       char(48),  /* Name of triggered queue */
```

```
  3 ProcessName char(48),  /* Name of process object */
  3 TriggerData char(64),  /* Trigger data */
  3 ApplType    char(4),   /* Application type */
  3 ApplId      char(256), /* Application identifier */
  3 EnvData     char(128), /* Environment data */
  3 UserData    char(128), /* User data */
  3 QMgrName    char(48);  /* Queue manager name */
```

Deklaration in High Level Assembler

```
MQTMC              DSECT
MQTMC_STRUCID      DS   CL4    Structure identifier
MQTMC_VERSION      DS   CL4    Structure version number
MQTMC_QNAME        DS   CL48   Name of triggered queue
MQTMC_PROCESSNAME  DS   CL48   Name of process object
MQTMC_TRIGGERDATA  DS   CL64   Trigger data
MQTMC_APPLTYPE     DS   CL4    Application type
MQTMC_APPLID       DS   CL256  Application identifier
MQTMC_ENVDATA      DS   CL128  Environment data
MQTMC_USERDATA     DS   CL128  User data
MQTMC_QMGRNAME     DS   CL48   Queue manager name
*
MQTMC_LENGTH       EQU  *-MQTMC
                   ORG  MQTMC
MQTMC_AREA         DS   CL(MQTMC_LENGTH)
```

Deklaration in Visual Basic

```
Type MQTMC2
  StrucId     As String*4   'Structure identifier'
  Version     As String*4   'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As String*4   'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
  QMgrName    As String*48  'Queue manager name'
End Type
```

MQWIH - Auslastungs-Header

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 557. Felder für MQWIH

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
StrucLength	Länge der MQWIH-Struktur	StrucLength
Encoding	Numerische Codierung der Daten, die auf MQWIH folgen	Codierung
CodedCharSetId	Zeichensatzkennung der Daten, die auf MQWIH folgen	CodedCharSetId
Format	Formatname der Daten, die auf MQWIH folgen	Format
Flags	Markierungen	Flags
ServiceName	Servicename	ServiceName
ServiceStep	Name der Servicestufe	ServiceStep

Tabelle 557. Felder für MQWIH (Forts.)		
Feld	**Beschreibung**	**Thema**
MsgToken	Nachrichtentoken	MsgToken
Reserved	Reserved	Reserved

Überblick zu MQWIH

Verfügbarkeit: Alle WebSphere MQ-Systeme sowie WebSphere MQ-Clients, die mit diesen Systemen verbunden sind.

Zweck: Die MQWIH-Struktur beschreibt die Informationen, die am Anfang der Nachricht vorhanden sein müssen, die vom z/OS Workload-Manager verarbeitet wird.

Formatname: MQFMT_WORK_INFO_HEADER.

Zeichensatz und Codierung: Die Felder in der MQWIH-Struktur haben den Zeichensatz und die Codierung, die durch die Felder *CodedCharSetId* und *Encoding* in der Headerstruktur vor MQWIH oder durch die Felder in der MQMD-Struktur angegeben werden, wenn sich MQWIH am Anfang der Anwendungsnachrichtendaten befindet.

Um in Warteschlangennamen gültige Zeichen bereitstellen zu können, muss der Zeichensatz Einzelbytezeichen unterstützen.

Verwendung: Wenn eine Nachricht durch den z/OS Workload-Manager verarbeitet werden soll, muss die Nachricht mit einer MQWIH-Struktur beginnen.

Felder für MQWIH

Die MQWIH-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

CodedCharSetId (MQLONG)

Gibt die Zeichensatzkennung der Daten an, die auf die MQWIH-Struktur folgen. Dieses Attribut bezieht sich nicht auf Zeichendaten in der MQWIH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Sie können den folgenden Spezialwert verwenden:

MQCCSI_INHERIT
Zeichendaten innerhalb der Daten, die auf diese Struktur *folgen* verwenden denselben Zeichensatz wie die Struktur.

Der Warteschlangenmanager ändert diesen Wert in der Struktur, die in der Nachricht mitgesendet wird, auf die tatsächliche Zeichensatz-ID der Struktur. Falls keine Fehler auftreten, wird der Wert MQCCSI_INHERIT nicht vom MQGET-Aufruf zurückgegeben.

MQCCSI_INHERIT kann nicht verwendet werden, wenn der Wert des Felds *PutApplType* in MQMD MQAT_BROKER ist.

Der Anfangswert dieses Felds lautet MQCCSI_UNDEFINED.

Encoding (MQLONG)

Gibt die numerische Codierung der Daten an, die auf die MQWIH-Struktur folgen. Dieses Attribut bezieht sich nicht auf numerische Daten in der MQWIH-Struktur selbst.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht.

Der Anfangswert dieses Feldes ist 0.

Flags (MQLONG)

Folgende Werte sind möglich:

MQWIH_NONE
Keine Flags.

Der Anfangswert dieses Felds ist MQWIH_NONE.

Format (MQCHAR8)

Dies gibt den Formatnamen der Daten an, die auf die MQWIH-Struktur folgen.

Im MQPUT- oder MQPUT1-Aufruf muss die Anwendung dieses Feld auf den Wert setzen, der den Daten entspricht. Die Regeln für die Codierung dieses Felds entsprechen denen für das Feld *Format* in MQMD.

Die Länge des Felds wird durch MQ_FORMAT_LENGTH angegeben. Der Anfangswert dieses Felds ist MQFMT_NONE.

MsgToken (MQBYTE16)

Dies ist das Nachrichtentoken, das die Nachricht eindeutig identifiziert.

Bei den MQPUT- und MQPUT1-Aufrufen wird dieses Feld ignoriert. Die Länge dieses Felds wird durch MQ_MSG_TOKEN_LENGTH angegeben. Der Anfangswert dieses Felds ist MQMTOK_NONE.

Reserved (MQCHAR32)

Dies ist ein reserviertes Feld; es muss leer sein.

ServiceName (MQCHAR32)

Dies ist der Name des Services, der die Nachricht verarbeiten soll.

Die Länge dieses Felds wird durch MQ_SERVICE_NAME_LENGTH angegeben. Der Anfangswert dieses Felds ist 32 Leerzeichen.

ServiceStep (MQCHAR8)

Gibt die Stufe von *ServiceName* an, auf die sich die Nachricht bezieht.

Die Länge dieses Felds wird durch MQ_SERVICE_STEP_LENGTH angegeben. Der Anfangswert dieses Felds ist 8 Leerzeichen.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQWIH_STRUC_ID
ID der Auslastungs-Headerstruktur

Für die Programmiersprache C ist auch die Konstante MQWIH_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQWIH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQWIH_STRUC_ID.

StrucLength (MQLONG)

Gibt die Länge der MQWIH-Struktur an. Folgende Werte sind möglich:

MQWIH_LENGTH_1
Länge der Auslastungs-Headerstruktur der Version 1

Die folgende Konstante gibt die Länge der aktuellen Version an:

MQWIH_CURRENT_LENGTH
Länge der aktuellen Version der Auslastungs-Headerstruktur

Der Anfangswert dieses Felds ist MQWIH_LENGTH_1.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQWIH_VERSION_1
Auslastungs-Headerstruktur der Version 1

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQWIH_CURRENT_VERSION
Aktuelle Version des Headers für Arbeitsinformationen.

Der Anfangswert dieses Felds ist MQWIH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQWIH

Tabelle 558. Anfangswerte für Felder im MQWIH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQWIH_STRUC_ID	'WIH¬'
Version	MQWIH_VERSION_1	1
StrucLength	MQWIH_LENGTH_1	120
Encoding	--	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Leerzeichen
Flags	MQWIH_NONE	0
ServiceName	--	Leerzeichen
ServiceStep	--	Leerzeichen
MsgToken	MQMTOK_NONE	Nullen
Reserved	--	Leerzeichen

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. In der Programmiersprache C enthält die MakrovariableMQWIH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQWIH structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQWIH */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQWIH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQWIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR32  ServiceName;     /* Service name */
  MQCHAR8   ServiceStep;     /* Service step name */
  MQBYTE16  MsgToken;        /* Message token */
  MQCHAR32  Reserved;        /* Reserved */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQWIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQWIH structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQWIH */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQWIH */
  3 Format         char(8),       /* Format name of data that follows
                                     MQWIH */
  3 Flags          fixed bin(31), /* Flags */
  3 ServiceName    char(32),      /* Service name */
  3 ServiceStep    char(8),       /* Service step name */
  3 MsgToken       char(16),      /* Message token */
  3 Reserved       char(32);      /* Reserved */
```

Deklaration in High Level Assembler

```
MQWIH                 DSECT
MQWIH_STRUCID         DS   CL4   Structure identifier
MQWIH_VERSION         DS   F     Structure version number
MQWIH_STRUCLENGTH     DS   F     Length of MQWIH structure
MQWIH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQWIH
MQWIH_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQWIH
MQWIH_FORMAT          DS   CL8   Format name of data that follows MQWIH
MQWIH_FLAGS           DS   F     Flags
MQWIH_SERVICENAME     DS   CL32  Service name
MQWIH_SERVICESTEP     DS   CL8   Service step name
MQWIH_MSGTOKEN        DS   XL16  Message token
MQWIH_RESERVED        DS   CL32  Reserved
*
MQWIH_LENGTH          EQU  *-MQWIH
                      ORG  MQWIH
MQWIH_AREA            DS   CL(MQWIH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQWIH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
```

```
  StrucLength    As Long      'Length of MQWIH structure'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQWIH'
  CodedCharSetId As Long      'Character-set identifier of data that'
                              'follows MQWIH'
  Format         As String*8  'Format name of data that follows MQWIH'
  Flags          As Long      'Flags'
  ServiceName    As String*32 'Service name'
  ServiceStep    As String*8  'Service step name'
  MsgToken       As MQBYTE16  'Message token'
  Reserved       As String*32 'Reserved'
End Type
```

MQXP - Exitparameterblock

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 559. Felder für MQXP		
Feld	**Beschreibung**	**Thema**
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
ExitId	Exit-ID	ExitId
ExitReason	Ursache für Aufruf des Exits	ExitReason
ExitResponse	Antwort vom Exit	ExitResponse
ExitCommand	API-Aufrufcode	ExitCommand
ExitParmCount	Parameteranzahl	ExitParmCount
ExitUserArea	Benutzerbereich	ExitUserArea

Überblick zu MQXP

Verfügbarkeit: z/OS.

Zweck: Die MQXP-Struktur wird als Ein-/Ausgabeparameter an den API-Steuerübergabeexit verwendet. Weitere Informationen zu diesem Exit finden Sie im Abschnitt API-Steuerübergabeexit.

Zeichensatz und Codierung: Der Zeichensatz der Zeichendaten in MQXP entspricht dem des lokalen Warteschlangenmanagers, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird. Numerische Daten in MQXP entsprechen der nativen Systemcodierung, die durch MQENC_NATIVE angegeben wird.

Felder für MQXP

Die MQXP-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

ExitCommand (MQLONG)

Dieses Feld wird beim Einstieg in die Exitroutine festgelegt. Es gibt den API-Aufruf an, der den Aufruf des Exits ausgelöst hat:

MQXC_CALLBACK
CALLBACK-Aufruf

MQXC_MQBACK
MQBACK-Aufruf

MQXC_MQCB
MQCB-Aufruf

MQXC_MQCLOSE
MQCLOSE-Aufruf

MQXC_MQCMIT
MQCMIT-Aufruf

MQXC_MQCTL
MQCTL-Aufruf

MQXC_MQGET
MQGET-Aufruf

MQXC_MQINQ
MQINQ-Aufruf

MQXC_MQOPEN
MQOPEN-Aufruf

MQXC_MQPUT
MQPUT-Aufruf

MQXC_MQPUT1
MQPUT1-Aufruf

MQXC_MQSET
MQSET-Aufruf

MQXC_MQSTAT
MQSTAT-Aufruf

MQXC_MQSUB
MQSUB-Aufruf

MQXC_MQSUBRQ
MQSUBRQ-Aufruf

Dies ist ein Eingabefeld für den Exit.

ExitId (MQLONG)

Dies wird beim Einstieg in die Exitroutine festgelegt und gibt den Typ des Exits an:

MQXT_API_CROSSING_EXIT
API-Steuerübergabeexit für CICS

Dies ist ein Eingabefeld für den Exit.

ExitParmCount (MQLONG)

Dieses Feld wird beim Einstieg in die Exitroutine festgelegt. Es enthält die Anzahl Parameter, die der MQ-Aufruf benötigt. Diese sind:

Name des Aufrufs	Anzahl der Parameter
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Dies ist ein Eingabefeld für den Exit.

ExitReason (MQLONG)

Dies wird beim Einstieg in die Exitroutine festgelegt. Beim API-Steuerübergabeexit gibt es an, ob die Routine vor oder nach dem API-Aufruf aufgerufen wird:

MQXR_BEFORE
Vor der API-Ausführung

MQXR_AFTER
Nach der API-Ausführung

Dies ist ein Eingabefeld für den Exit.

ExitResponse (MQLONG)

Der Wert wird vom Exit festgelegt, um mit dem aufrufenden Programm zu kommunizieren. Die folgenden Werte sind definiert:

MQXCC_OK
Exit erfolgreich ausgeführt.

MQXCC_SUPPRESS_FUNCTION
Funktion unterdrücken.

Wenn dieser Wert durch einen API-Steuerübergabeexit festgelegt wird, der *vor* dem API-Aufruf aufgerufen wird, wird der API-Aufruf nicht durchgeführt. *CompCode* für diesen Aufruf wird auf MQCC_FAILED und *Reason* wird auf MQRC_SUPPRESSED_BY_EXIT festgelegt, alle anderen Parameter verbleiben so, wie der Exit sie festgelegt hat.

Wenn dieser Wert durch einen API-Steuerübergabeexit festgelegt wird, der *nach* dem API-Aufruf aufgerufen wird, wird er vom Warteschlangenmanager ignoriert.

MQXCC_SKIP_FUNCTION
Sprungfunktion.

Wenn dieser Wert durch einen API-Steuerübergabeexit festgelegt wird, der *vor* dem API-Aufruf aufgerufen wird, wird der API-Aufruf nicht durchgeführt; die Parameter *CompCode* und *Reason* sowie alle anderen Parameter verbleiben so, wie der Exit sie festgelegt hat.

Wenn dieser Wert durch einen API-Steuerübergabeexit festgelegt wird, der *nach* dem API-Aufruf aufgerufen wird, wird er vom Warteschlangenmanager ignoriert.

Dies ist ein Ausgabefeld vom Exit.

ExitUserArea (MQBYTE16)

Dies ist ein Feld, das dem Exit zur Verwendung verfügbar ist. Es wird auf binäre Null für die Feldlänge vor dem ersten Aufruf des Exits für die Aufgabe initialisiert, danach werden Änderungen, die der Exit an diesem Feld durchführt, für alle Aufrufe des Exits beibehalten. Der folgende Wert ist definiert:

MQXUA_NONE
Keine Benutzerinformationen.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQXUA_NONE_ARRAY definiert. Sie hat den gleichen Wert wie MQXUA_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_EXIT_USER_AREA_LENGTH angegeben. Dies ist ein Ein-/Ausgabefeld für den Exit.

Reserved (MQLONG)

Dies ist ein reserviertes Feld. Der Wert ist für den Exit nicht signifikant.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQXP_STRUC_ID
ID für die Exitparameterstruktur

Für die Programmiersprache C ist auch die Konstante MQXP_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQXP_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Eingabefeld für den Exit.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQXP_VERSION_1
Versionsnummer für die Exitparameterblockstruktur

Anmerkung: Wenn eine neue Version dieser Struktur eingeführt wird, wird das Layout des vorhandenen Teils nicht geändert. Der Exit muss daher überprüfen, dass die Versionsnummer gleich oder größer als die niedrigste Version ist, die die Felder enthält, die der Exit verwenden muss.

Dies ist ein Eingabefeld für den Exit.

Sprachendeklarationen

Diese Struktur wird in den folgenden Programmiersprachen unterstützt.

Deklaration in Programmiersprache C

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;         /* Exit identifier */
  MQLONG    ExitReason;     /* Reason for invocation of exit */
  MQLONG    ExitResponse;   /* Response from exit */
  MQLONG    ExitCommand;    /* API call code */
  MQLONG    ExitParmCount;  /* Parameter count */
  MQLONG    Reserved;       /* Reserved */
  MQBYTE16  ExitUserArea;   /* User area */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQXP based,
```

```
  3 StrucId       char(4),       /* Structure identifier */
  3 Version       fixed bin(31), /* Structure version number */
  3 ExitId        fixed bin(31), /* Exit identifier */
  3 ExitReason    fixed bin(31), /* Reason for invocation of exit */
  3 ExitResponse  fixed bin(31), /* Response from exit */
  3 ExitCommand   fixed bin(31), /* API call code */
  3 ExitParmCount fixed bin(31), /* Parameter count */
  3 Reserved      fixed bin(31), /* Reserved */
  3 ExitUserArea  char(16);      /* User area */
```

Deklaration in High Level Assembler

```
MQXP                DSECT
MQXP_STRUCID        DS   CL4   Structure identifier
MQXP_VERSION        DS   F     Structure version number
MQXP_EXITID         DS   F     Exit identifier
MQXP_EXITREASON     DS   F     Reason for invocation of exit
MQXP_EXITRESPONSE   DS   F     Response from exit
MQXP_EXITCOMMAND    DS   F     API call code
MQXP_EXITPARMCOUNT  DS   F     Parameter count
MQXP_RESERVED       DS   F     Reserved
MQXP_EXITUSERAREA   DS   XL16  User area
*
MQXP_LENGTH         EQU  *-MQXP
                    ORG  MQXP
MQXP_AREA           DS   CL(MQXP_LENGTH)
```

MQXQH – Header der Übertragungswarteschlange

In der folgenden Tabelle werden die Felder in der Struktur zusammengefasst.

Tabelle 560. Felder für MQXQH

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
RemoteQName	Name der Zielwarteschlange	RemoteQName
RemoteQMgrName	Name des Ziel-Warteschlangenmanagers	RemoteQMgrName
MsgDesc	Ursprünglicher Nachrichtendeskriptor	MsgDesc

Überblick zu MQXQH

Verfügbarkeit: Alle WebSphere MQ-Systeme und WebSphere MQ-Clients.

Zweck: Die MQXQH-Struktur beschreibt die Informationen, die den Anwendungsnachrichtendaten von Nachrichten in Übertragungswarteschlangen vorangestellt werden. Eine Übertragungswarteschlange ist eine spezielle lokale Warteschlange, die temporär Nachrichten aufnimmt, die für ferne Warteschlangen bestimmt sind (d. h. für Warteschlangen, deren Eigner nicht der lokale Warteschlangenmanager ist). Eine Übertragungswarteschlange wird durch das Warteschlangenattribut *Usage* mit dem Wert MQUS_TRANSMISSION angegeben.

Formatname: MQFMT_XMIT_Q_HEADER.

Zeichensatz und Codierung: Der Zeichensatz von Daten in MQXOH muss dem entsprechen, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben ist, die Codierung der des lokalen Warteschlangenmanagers, die durch MQENC_NATIVE angebeben ist.

Definieren Sie den Zeichensatz und die Codierung von MQXQH in den Feldern *CodedCharSetId* und *Encoding* folgendermaßen:

- Im separaten MQMD (wenn sich die MQXQH-Struktur am Anfang der Nachrichtendaten befindet) oder
- In der Headerstruktur, die der MQXQH-Struktur vorangeht (alle anderen Fälle).

Verwendung: Eine Nachricht in einer Übertragungswarteschlange hat *zwei* Nachrichtendeskriptoren:

- Ein Nachrichtendeskriptor wird getrennt von den Nachrichtendaten gespeichert. Dies ist der sogenannte *separate Nachrichtendeskriptor*, der durch den Warteschlangenmanager generiert wird, wenn die Nachricht in die Übertragungswarteschlange eingereiht wird. Einige Felder im separaten Nachrichtendeskriptor werden aus dem Nachrichtendeskriptor kopiert, der durch die Anwendung beim MQPUT- oder MQPUT1-Aufruf bereitgestellt wird.

  Der separate Nachrichtendeskriptor wird an die Anwendung im Parameter *MsgDesc* des MQGET-Aufrufs zurückgegeben, wenn die Nachricht aus der Übertragungswarteschlange entfernt wird.
- Ein zweiter Nachrichtendeskriptor wird innerhalb der MQXQH-Struktur als Teil der Nachrichtendaten gespeichert, dies ist der *eingebettete Nachrichtendeskriptor*. Er ist eine Kopie des Nachrichtendeskriptors, der von der Anwendung beim MQPUT- oder MQPUT1-Aufruf (mit geringfügigen Änderungen) bereitgestellt wird.

  Der eingebettete Nachrichtendeskriptor ist immer ein MQMD der Version 1. Wenn die durch die Anwendung eingereihte Nachricht für eines oder mehrere der Felder der Version 2 im MQMD Werte hat, die keine Standardwerte sind, dann folgt dem MQXQH eine MQMDE-Struktur, der wiederum die Anwendungsnachrichtendaten folgen (sofern vorhanden). Die MQMDE wird entweder:

  – durch den Warteschlangenmanager generiert (wenn die Anwendung für das Einreihen der Nachricht einen MQMD der Version 2 verwendet) oder
  – ist bereits am Anfang der Anwendungsnachrichtendaten vorhanden (wenn die Anwendung für das Einreihen der Nachricht einen MQMD der Version 1 verwendet).

  Der eingebettete Nachrichtendeskriptor wird an die Anwendung im Parameter *MsgDesc* des MQGET-Aufrufs zurückgegeben, wenn die Nachricht aus der endgültigen Zielwarteschlange entfernt wird.

Felder im separaten Nachrichtendeskriptor: Die Felder im separaten Nachrichtendeskriptor werden wie dargestellt vom Warteschlangenmanager festgelegt. Wenn der Warteschlangenmanager MQMD der Version 2 nicht unterstützt, wird ohne Funktionseinbußen ein MQMD der Version 1 verwendet.

Feld in separatem MQMD	Verwendeter Wert
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	Aus dem eingebetteten Nachrichtendeskriptor kopiert, wobei die durch MQRO_ACCEPT_UNSUP_IF_XMIT_MASK identifizierten Bits auf null gesetzt werden (Dadurch wird verhindert, dass beim Einreihen oder Entfernen einer Nachricht in eine bzw. aus einer Übertragungswarteschlange eine COA- oder COD-Berichtsnachricht generiert wird.)
MsgType	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
Expiry	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
Feedback	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
Encoding	MQENC_NATIVE (siehe Hinweis)
CodedCharSetId	Attribut *CodedCharSetId* des Warteschlangenmanagers.
Format	MQFMT_XMIT_Q_HEADER
Priority	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
Persistence	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
MsgId	Durch den Warteschlangenmanager wird ein neuer Wert generiert. Diese Nachrichten-ID unterscheidet sich von der *MsgId*, die der Warteschlangenmanager möglicherweise für den eingebettete Nachrichtendeskriptor generiert hat (siehe oben).

Feld in separatem MQMD	Verwendeter Wert
CorrelId	Die *MsgId* vom eingebetteten Nachrichtendeskriptor. Bei Nachrichten, die in SYSTEM.CLUSTER.TRANSMIT.QUEUE eingereiht werden, ist *CorrelId* für die interne Verwendung reserviert.
BackoutCount	0
ReplyToQ	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
ReplyToQMgr	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
UserIdentifier	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
AccountingToken	Aus dem eingebetteten Nachrichtendeskriptor kopiert. Bei Nachrichten, die in SYSTEM.CLUSTER.TRANSMIT.QUEUE eingereiht werden, ist *AccountingToken* für die interne Verwendung reserviert.
ApplIdentityData	Aus dem eingebetteten Nachrichtendeskriptor kopiert.
PutApplType	MQAT_QMGR
PutApplName	Erste 28 Byte des Warteschlangenmanagernamens
PutDate	Datum, an dem die Nachricht in die Übertragungswarteschlange eingereiht wurde
PutTime	Uhrzeit, zu der die Nachricht in die Übertragungswarteschlange eingereiht wurde.
ApplOriginData	Leerzeichen
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- Unter Windowsweicht der Wert von MQENC_NATIVE für Micro Focus COBOL vom Wert für C ab. Der Wert im Feld *Encoding* im separaten Nachrichtendeskriptor ist immer der Wert für C in diesen Umgebungen. Dieser Wert ist 546 im Dezimalformat. Auch die Codierung der Ganzzahlfelder in der MQXQH-Struktur entspricht diesem Wert (die native Intel-Codierung).

Felder im eingebetteten Nachrichtendeskriptor: Die Felder im eingebetteten Nachrichtendeskriptor haben dieselben Werte wie der Parameter *MsgDesc* des MQPUT- oder MQPUT1-Aufrufs, mit folgenden Ausnahmen:

- Das Feld *Version* hat immer den Wert MQMD_VERSION_1.
- Wenn das Feld *Priority* den Wert MQPRI_PRIORITY_AS_Q_DEF hat, wird er durch den Wert des Warteschlangenattributs *DefPriority* ersetzt.
- Wenn das Feld *Persistence* den Wert MQPER_PERSISTENCE_AS_Q_DEF hat, wird er durch den Wert des Warteschlangenattributs *DefPersistence* ersetzt.
- Wenn das Feld *MsgId* den Wert MQMI_NONE hat, wenn die Option MQPMO_NEW_MSG_ID angegeben wurde oder wenn die Nachricht eine Verteilerlistennachricht ist, wird *MsgId* durch eine neue, vom Warteschlangenmanager generierte Nachrichten-ID ersetzt.

  Wenn eine Verteilerlistennachricht in kleinere Verteilerlistennachrichten aufgeteilt und in verschiedene Übertragungswarteschlangen platziert wird, ist das Feld *MsgId* in jedem neuen eingebetteten Nachrichtendeskriptor dasselbe wie in der ursprünglichen Verteilerlistennachricht.
- Wenn die Option MQPMO_NEW_CORREL_ID angegeben wurde, wird *CorrelId* durch eine neue, vom Warteschlangenmanager generierte Korrelations-ID ersetzt.

- Die Kontextfelder werden entsprechend den Optionen MQPMO_*_CONTEXT festgelegt, die im Parameter *PutMsgOpts* angegeben werden. Kontextfelder sind:
  - *AccountingToken*
  - *ApplIdentityData*
  - *ApplOriginData*
  - *PutApplName*
  - *PutApplType*
  - *PutDate*
  - *PutTime*
  - *UserIdentifier*
- Die Felder der Version 2 (falls solche vorhanden waren) werden aus dem MQMD entfernt und in eine MQMDE-Struktur verschoben, wenn mindestens eines der Felder der Version 2 auf einen anderen Wert als den Standardwert festgelegt ist.

Nachrichten in ferne Warteschlangen einreihen: Wenn eine Anwendung eine Nachricht in eine ferne Warteschlange einreiht (indem entweder der Name der fernen Warteschlange direkt angegeben oder eine lokale Definition der fernen Warteschlange verwendet wird), führt der lokale Warteschlangenmanager Folgendes durch:

- Er erstellt eine MQXQH-Struktur, die den eingebetteten Nachrichtendeskriptor enthält
- Er fügt eine MQMDE an, wenn sie benötigt wird und nicht vorhanden ist
- Er fügt die Anwendungsnachrichtendaten an
- Er platziert die Nachricht in eine entsprechende Übertragungswarteschlange

Nachrichten direkt in Übertragungswarteschlangen einreihen: Eine Anwendung kann auch eine Nachricht direkt in eine Übertragungswarteschlange einreihen. In diesem Fall muss die Anwendung den Anwendungsnachrichtendaten eine MQXQH-Struktur voranstellen und die Felder mit den passenden Werten initialisieren. Zusätzlich muss das Feld *Format* im Parameter *MsgDesc* des MQPUT- oder MQPUT1-Aufrufs den Wert MQFMT_XMIT_Q_HEADER haben.

Der Zeichensat der Zeichendaten in der MQXQH-Struktur, die von der Anwendung erstellt werden, muss dem des lokalen Warteschlangenmanagers entsprechen (festgelegt durch das Warteschlangenmanagerattribut *CodedCharSetId*), ganzzahlige Daten müssen der nativen Systemcodierung entsprechen. Außerdem müssen Zeichendaten in der MQXQH-Struktur mit Leerzeichen auf die definierte Länge des Felds aufgefüllt werden. Die Daten dürfen nicht vorzeitig durch das Nullzeichen beendet werden, weil der Warteschlangenmanager die Null und nachfolgende Zeichen in der MQXQH-Struktur nicht konvertiert.

Der Warteschlangenmanager überprüft jedoch nicht, ob die MQXQH-Struktur vorhanden ist oder gültige Werte für die Felder angegeben wurden.

Anwendungen dürfen Nachrichten nicht direkt in SYSTEM.CLUSTER.TRANSMIT.QUEUE einreihen.

Nachrichten aus Übertragungswarteschlangen abrufen: Anwendungen, die Nachrichten aus einer Übertragungswarteschlange abrufen, müssen die Informationen in der MQXQH-Struktur in entsprechender Weise verarbeiten. Das Vorhandensein der MQXQH-Struktur am Anfang der Anwendungsnachrichtendaten wird durch den Wert MQFMT_XMIT_Q_HEADER angegeben, der im Feld *Format* im Parameter *MsgDesc* des MQGET-Aufrufs zurückgegeben wird. Die Werte, die in den Feldern *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* zurückgegeben werden, geben den Zeichensatz und die Codierung der Zeichen und der ganzzahligen Daten in der MQXQH-Struktur an. Der Zeichensatz und die Codierung der Anwendungsnachrichtendaten werden durch die Felder *CodedCharSetId* und *Encoding* im eingebetteten Nachrichtendeskriptor festgelegt.

Felder für MQXQH

Die MQXQH-Struktur umfasst die folgenden Felder, sie werden in **alphabetischer Reihenfolge** beschrieben:

MsgDesc (MQMD1)

Dies ist der eingebettete Nachrichtendeskriptor, eine genaue Kopie des Nachrichtendeskriptors MQMD, der als Parameter *MsgDesc* beim MQPUT- oder MQPUT1-Aufruf angegeben wurde, als die Nachricht ursprünglich in die ferne Warteschlange eingereiht wurde.

Anmerkung: Dies ist ein MQMD der Version 1.

Die Anfangswerte der Felder in der Struktur sind dieselben wie in der MQMD-Struktur.

RemoteQMgrName (MQCHAR48)

Dies ist der Name des Warteschlangenmanagers oder der Gruppe mit gemeinsamer Warteschlange, der bzw. die Eigner der das anscheinende endgültige Ziel für die Nachricht darstellenden Warteschlange ist.

Wenn die Nachricht eine Verteilerlistennachricht ist, ist *RemoteQMgrName* leer.

Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

RemoteQName (MQCHAR48)

Dies ist der Name der Nachrichtenwarteschlange, die offenbar das endgültige Ziel für die Nachricht ist (dies ist möglicherweise nicht das endgültige Ziel, wenn diese Warteschlange beispielsweise bei *RemoteQMgrName* als lokale Definition einer anderen fernen Warteschlange festgelegt ist).

Wenn die Nachricht eine Verteilerlistennachricht ist (d. h. das Feld *Format* im eingebetteten Nachrichtendeskriptor lautet MQFMT_DIST_HEADER), ist *RemoteQName* leer.

Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben. Der Anfangswert dieses Feldes ist die Nullzeichenfolge in C und 48 leere Zeichen in anderen Programmiersprachen.

StrucId (MQCHAR4)

Dies ist die Struktur-ID. Folgende Werte sind möglich:

MQXQH_STRUC_ID
ID für die Headerstruktur der Übertragungswarteschlange

Für die Programmiersprache C ist auch die Konstante MQXQH_STRUC_ID_ARRAY definiert. Sie hat den gleichen Wert wie die MQXQH_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQXQH_STRUC_ID.

Version (MQLONG)

Dies ist die Strukturversionsnummer. Folgende Werte sind möglich:

MQXQH_VERSION_1
Versionsnummer für die Headerstruktur der Übertragungswarteschlange

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQXQH_CURRENT_VERSION
Aktuelle Version der Struktur des Headers für die Übertragungswarteschlange.

Der Anfangswert dieses Felds ist MQXQH_VERSION_1.

Anfangswerte und Sprachendeklarationen für MQXQH

Tabelle 561. Anfangswerte für Felder im MQXQH

Name des Felds	Name der Konstante	Wert der Konstanten
StrucId	MQXQH_STRUC_ID	'XQH¬'
Version	MQXQH_VERSION_1	1

Tabelle 561. Anfangswerte für Felder im MQXQH (Forts.)		
Name des Felds	**Name der Konstante**	**Wert der Konstanten**
RemoteQName	--	Nullzeichenfolge oder Leerzeichen.
RemoteQMgrName	--	Nullzeichenfolge oder Leerzeichen.
MsgDesc	Gleiche Namen und Werte wie beim MQMD; siehe Tabelle 515 auf Seite 449	-

Anmerkungen:

1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar.
2. Der Wert "Nullzeichenfolge" oder "Leerzeichen" kennzeichnet die Nullzeichenfolge in C und Leerzeichen in anderen Programmiersprachen.
3. In der Programmiersprache C enthält die MakrovariableMQXQH_DEFAULT die oben aufgelisteten Werte. Verwenden Sie sie folgendermaßen, um Anfangswerte für die Felder in der Struktur anzugeben:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Deklaration in Programmiersprache C

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQCHAR48  RemoteQName;     /* Name of destination queue */
  MQCHAR48  RemoteQMgrName;  /* Name of destination queue manager */
  MQMD1     MsgDesc;         /* Original message descriptor */
};
```

COBOL-Delaration

```
**   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).
```

Deklaration in PL/I

```
dcl
 1 MQXQH based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 RemoteQName       char(48),      /* Name of destination queue */
  3 RemoteQMgrName    char(48),      /* Name of destination queue
                                        manager */
  3 MsgDesc,                         /* Original message descriptor */
   5 StrucId          char(4),       /* Structure identifier */
   5 Version          fixed bin(31), /* Structure version number */
   5 Report           fixed bin(31), /* Report options */
   5 MsgType          fixed bin(31), /* Message type */
   5 Expiry           fixed bin(31), /* Expiry time */
   5 Feedback         fixed bin(31), /* Feedback or reason code */
   5 Encoding         fixed bin(31), /* Numeric encoding of message
                                        data */
   5 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                        message data */
   5 Format           char(8),       /* Format name of message data */
   5 Priority         fixed bin(31), /* Message priority */
   5 Persistence      fixed bin(31), /* Message persistence */
   5 MsgId            char(24),      /* Message identifier */
   5 CorrelId         char(24),      /* Correlation identifier */
   5 BackoutCount     fixed bin(31), /* Backout counter */
   5 ReplyToQ         char(48),      /* Name of reply-to queue */
   5 ReplyToQMgr      char(48),      /* Name of reply queue manager */
   5 UserIdentifier   char(12),      /* User identifier */
   5 AccountingToken  char(32),      /* Accounting token */
   5 ApplIdentityData char(32),      /* Application data relating to
                                        identity */
   5 PutApplType      fixed bin(31), /* Type of application that put the
                                        message */
   5 PutApplName      char(28),      /* Name of application that put the
                                        message */
   5 PutDate          char(8),       /* Date when message was put */
   5 PutTime          char(8),       /* Time when message was put */
   5 ApplOriginData   char(4);       /* Application data relating to
                                        origin */
```

Deklaration in High Level Assembler

```
MQXQH                           DSECT
MQXQH_STRUCID                   DS   CL4   Structure identifier
```

```
MQXQH_VERSION                   DS   F     Structure version number
MQXQH_REMOTEQNAME               DS   CL48  Name of destination queue
MQXQH_REMOTEQMGRNAME            DS   CL48  Name of destination queue
*                                          manager
MQXQH_MSGDESC                   DS   0F    Force fullword alignment
MQXQH_MSGDESC_STRUCID           DS   CL4   Structure identifier
MQXQH_MSGDESC_VERSION           DS   F     Structure version number
MQXQH_MSGDESC_REPORT            DS   F     Report options
MQXQH_MSGDESC_MSGTYPE           DS   F     Message type
MQXQH_MSGDESC_EXPIRY            DS   F     Expiry time
MQXQH_MSGDESC_FEEDBACK          DS   F     Feedback or reason code
MQXQH_MSGDESC_ENCODING          DS   F     Numeric encoding of message
*                                          data
MQXQH_MSGDESC_CODEDCHARSETID    DS   F     Character set identifier of
*                                          message data
MQXQH_MSGDESC_FORMAT            DS   CL8   Format name of message data
MQXQH_MSGDESC_PRIORITY          DS   F     Message priority
MQXQH_MSGDESC_PERSISTENCE       DS   F     Message persistence
MQXQH_MSGDESC_MSGID             DS   XL24  Message identifier
MQXQH_MSGDESC_CORRELID          DS   XL24  Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT      DS   F     Backout counter
MQXQH_MSGDESC_REPLYTOQ          DS   CL48  Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR       DS   CL48  Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER    DS   CL12  User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN   DS   XL32  Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA  DS   CL32  Application data relating to
*                                          identity
MQXQH_MSGDESC_PUTAPPLTYPE       DS   F     Type of application that put
*                                          the message
MQXQH_MSGDESC_PUTAPPLNAME       DS   CL28  Name of application that put
*                                          the message
MQXQH_MSGDESC_PUTDATE           DS   CL8   Date when message was put
MQXQH_MSGDESC_PUTTIME           DS   CL8   Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA    DS   CL4   Application data relating to
*                                          origin
MQXQH_MSGDESC_LENGTH            EQU  *-MQXQH_MSGDESC
                                ORG  MQXQH_MSGDESC
MQXQH_MSGDESC_AREA              DS   CL(MQXQH_MSGDESC_LENGTH)
*
MQXQH_LENGTH                    EQU  *-MQXQH
                                ORG  MQXQH
MQXQH_AREA                      DS   CL(MQXQH_LENGTH)
```

Deklaration in Visual Basic

```
Type MQXQH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  RemoteQName    As String*48 'Name of destination queue'
  RemoteQMgrName As String*48 'Name of destination queue manager'
  MsgDesc        As MQMD1     'Original message descriptor'
End Type
```

Funktionsaufrufe

Dieser Abschnitt enthält Informationen zu allen MQI-Aufrufen, die möglich sind. Zu jedem der Aufrufe finden Sie Beschreibungen, die Syntax, Parameterinformationen, Hinweise zur Verwendung und Aufrufe für jede mögliche Programmiersprache.

Aufrufbeschreibungen

In diesem Abschnitt werden MQI-Aufrufe beschrieben.

- „MQBACK - Änderungen zurücksetzen” auf Seite 619
- „MQBEGIN - Arbeitseinheit starten” auf Seite 623
- „MQBUFMH - Konvertieren von Puffern in Nachrichtenkennungen” auf Seite 626
- „MQCB – Callback verwalten” auf Seite 630
- „MQCB_FUNCTION - Callback-Funktion” auf Seite 640
- „MQCLOSE - Objekt schließen” auf Seite 642

- „MQCMIT - Änderungen festschreiben” auf Seite 651
- „MQCONN - Warteschlangenmanager verbinden” auf Seite 655
- „MQCONNX – Verbindung mit Warteschlangenmanager herstellen (erweitert)” auf Seite 663
- „MQCRTMH - Nachrichtenkennung erstellen” auf Seite 669
- „MQCTL - Callbacks steuern” auf Seite 672
- „MQDISC - Verbindung mit Warteschlangenmanager beenden” auf Seite 678
- „MQDLTMH - Nachrichtenkennung löschen” auf Seite 682
- „MQDLTMP - Löschen von Nachrichteneigenschaften” auf Seite 685
- „MQGET - Nachricht abrufen” auf Seite 687
- „MQINQ - Objektattribute abfragen” auf Seite 700
- „MQINQMP - Abfragen von Nachrichteneigenschaften” auf Seite 718
- „MQMHBUF - Konvertieren von Nachrichtenkennungen in Puffer” auf Seite 724
- „MQOPEN – Objekt öffnen” auf Seite 728
- „MQPUT - Nachricht einreihen” auf Seite 747
- „MQPUT1 - Eine einzelne Nachricht einreihen” auf Seite 761
- „MQSET - Objektattribute festlegen” auf Seite 772
- „MQSETMP - Nachrichteneigenschaft festlegen” auf Seite 778
- „MQSTAT - Statusinformationen abrufen” auf Seite 782
- „MQMHBUF - Konvertieren von Nachrichtenkennungen in Puffer” auf Seite 724
- „MQSUB – Subskription registrieren” auf Seite 786
- „MQSUBRQ - Subskriptionsanforderung” auf Seite 794

Für diese Aufrufe ist auf den UNIX-Plattformen Onlinehilfe als *Man*-Pages verfügbar.

Anmerkung: Die Aufrufe für Datenkonvertierung, MQXCNVC und MQ_DATA_CONV_EXIT, finden Sie unter „Datenkonvertierungsexit” auf Seite 908.

In den Aufrufbeschreibungen verwendete Konventionen

Diese Themensammlung enthält für jeden Aufruf eine Beschreibung der Parameter und der Verwendung des Aufrufs in einem Format, das von der Programmiersprache unabhängig ist. Anschließend folgen typische Beispiele des Aufrufs sowie typische Deklarationen der Parameter in jeder der unterstützten Programmiersprachen.

Wichtig: Bei der Codierung von WebSphere MQ-API-Aufrufen müssen Sie sicherstellen, dass alle relevanten Parameter (wie in den folgenden Abschnitten beschrieben) angegeben werden. Andernfalls kann es zu unvorhersehbaren Ergebnissen kommen.

Die Beschreibung der Aufrufe enthält folgende Abschnitte:

Name des Aufrufs
Der Name des Aufrufs gefolgt von einer Kurzbeschreibung des Aufrufzwecks.

Parameter
Hinter dem Namen jedes Parameters steht der Datentyp in runden Klammern sowie eine der folgenden Angaben:

Eingabe
Sie übergeben Informationen im Parameter, wenn Sie den Aufruf ausführen.

Ausgabe
Der Warteschlangenmanager gibt Informationen im Parameter zurück, wenn der Aufruf beendet oder fehlgeschlagen ist.

Eingabe/Ausgabe
Sie übergeben Informationen im Parameter, wenn Sie den Aufruf ausführen, und der Warteschlangenmanager ändert die Informationen, wenn der Aufruf beendet oder fehlgeschlagen ist.

Beispiel:

Compcode (MQLONG) - Ausgabe

In einigen Fällen ist der Datentyp eine Struktur. Für alle Vorgänge finden Sie im Abschnitt „Elementardatentypen" auf Seite 218 mehr Informationen zum Datentyp bzw. zur Struktur.

Bei den letzten beiden Parametern in jedem Aufruf handelt es sich um einen Beendigungscode und einen Ursachencode. Der Beendigungscode gibt an, ob der Aufruf erfolgreich, teilweise oder überhaupt nicht abgeschlossen wurde. Weitere Informationen zur teilweisen Ausführung oder zum Fehlschlag des Aufrufs erhalten Sie mit dem Ursachencode. Weitere Informationen zu den einzelnen Beendigungs- und Ursachencodes finden Sie im Abschnitt „Rückgabecodes" auf Seite 873.

Hinweise zur Verwendung
Zusätzliche Informationen zu dem Aufruf, in denen seine Verwendung sowie eventuelle Nutzungseinschränkungen beschrieben werden.

Aufruf in Assembler
Typisches Beispiel für den Aufruf und die Deklaration der Parameter in der Assemblersprache.

C-Aufruf
Typisches Beispiel für den Aufruf und die Deklaration der Parameter in der Programmiersprache C.

Aufruf in COBOL
Typisches Beispiel für den Aufruf und die Deklaration der Parameter in COBOL.

Aufruf in PL/I
Typisches Beispiel für den Aufruf und die Deklaration der Parameter in PL/I.

Alle Parameter werden in der Pass-by-Reference-Methode übergeben.

Aufruf in Visual Basic
Typisches Beispiel für den Aufruf und die Deklaration der Parameter in Visual Basic.

Weitere Konventionen der Schreibweise:

Konstanten
Namen von Konstanten werden in Großbuchstaben angegeben, z. B. MQOO_OUTPUT. Eine Gruppe von Konstanten mit dem gleichen Präfix wird wie folgt angezeigt: MQIA_*. Siehe „Konstanten" auf Seite 51 für den Wert einer Konstanten.

Arrays
In einigen Aufrufen bestehen Parameter aus Arrays mit Zeichenfolgen ohne feste Größen. In den Beschreibungen dieser Parameter steht der Kleinbuchstabe n für eine numerische Konstante. Ersetzen Sie beim Codieren der Deklaration für einen solchen Parameter das n durch den erforderlichen numerischen Wert.

Verwenden der Aufrufe in der Programmiersprache C

Parameter, die *nur Eingabe* des Typs MQHCONN, MQHOBJ, MQHMSG oder MQLONG sind, werden als Wert übergeben. Für alle anderen Parameter wird der Parameter *Adresse* des Parameters nach Wert übergeben.

Sie müssen nicht alle Parameter, die als Adresse übergeben werden, jedes Mal angeben, wenn Sie eine Funktion aufrufen. Wenn Sie keinen bestimmten Parameter benötigen, geben Sie als Parameter beim Funktionsaufruf einen Nullzeiger anstatt der Adresse des Parameters an. Parameter, bei denen dies möglich ist, sind in den Aufrufbeschreibungen angegeben.

Kein Parameter wird als der Wert des Aufrufs zurückgegeben: in der C-Terminologie bedeutet dies, dass alle Aufrufe `void` zurückgeben.

Deklarieren der Pufferparameter

Beim MQGET-, MQPUT- und MQPUT1-Aufruf gibt es jeweils einen Parameter mit nicht definiertem Datentyp: den Parameter *Buffer*. Verwenden Sie diesen Parameter, um die Nachrichtendaten der Anwendung zu senden und zu empfangen.

Parameter dieser Art werden in den C-Beispielen als Arrays von MQBYTE dargestllt. Sie können die Parameter zwar auf diese Weise deklarieren, in der Regel ist es jedoch praktischer, sie als spezielle Struktur zu deklarieren, die den Aufbau der Daten in der Nachricht beschreibt. Der Funktionsprototyp deklariert den Parameter als Void-Zeiger, damit Sie die Adresse von beliebigen Datenarten als Parameter beim Aufruf angeben können.

Der Void-Zeiger ist ein Zeiger auf Daten mit nicht definiertem Format. Er wird folgendermaßen definiert:

```
typedef void *PMQVOID;
```

MQBACK - Änderungen zurücksetzen

Der MQBACK-Aufruf teilt dem Warteschlangenmanager mit, dass alle seit dem letzten Synchronisationspunkt vorgenommenen Nachrichteneinreihungen und -abrufe zurückgesetzt werden sollen.

Als Teil einer Arbeitseinheit eingereihte Nachrichten werden gelöscht; als Teil einer Arbeitseinheit abgerufene Nachrichten werden in der Warteschlange wiederhergestellt.

- Unter z/OS wird dieser Aufruf nur von Stapelverarbeitungsprogrammen verwendet (einschließlich IMS Stapel-DL/I-Programme).
- Unter IBM i wird dieser Aufruf nicht für Anwendungen unterstützt, die im Kompatibilitätsmodus ausgeführt werden.

Syntax

MQBACK (*Hconn*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Compcode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Aufruf in Umgebung nicht gültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OUTCOME_MIXED
(2123, X'84B') Ergebnis der Festschreibungs- oder Rücksetzungsoperation ist gemischt

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Sie können diesen Aufruf nur verwenden, wenn der Warteschlangenmanager selbst die Arbeitseinheit koordiniert. Dieser kann Folgendes einschließen:
   - Eine lokale Arbeitseinheit, bei der die Änderungen nur MQ-Ressourcen betreffen.
   - Eine globale Arbeitseinheit, bei der die Änderungen neben den MQ-Ressourcen auch Ressourcen anderer Ressourcenmanager betreffen können.

   Nähere Details über lokale und globale Arbeitseinheiten finden Sie im Abschnitt „MQBEGIN - Arbeitseinheit starten” auf Seite 623.
2. Verwenden Sie in Umgebungen, in denen der Warteschlangenmanager die Arbeitseinheit nicht koordiniert, den entsprechenden Aufruf zum Zurücksetzen anstelle von MQBACK. Die Umgebung unterstützt möglicherweise auch eine implizite Rücksetzung, die durch fehlerhaftes Beenden der Anwendung verursacht wird.
   - Unter z/OS verwenden Sie die folgenden Aufrufe:
     – Stapelverarbeitungsprogramme (einschließlich IMS Stapel-DL/I-Programme) können den MQBACK-Aufruf verwenden, wenn sich die Arbeitseinheit nur auf MQ-Ressourcen auswirkt. Wenn sich die Arbeitseinheit allerdings sowohl auf MQ-Ressourcen als auch auf Ressourcen anderer Resourcenmanager (z. B. DB2) auswirkt, verwenden Sie den SRRBACK-Aufruf, der vom z/OS Recoverable Resource Service (RRS) bereitgestellt wird. Der SRRBACK-Aufruf setzt Änderungen an

Ressourcen zurück, die zu den Resource Managers gehören, die für die RRS-Koordination aktiviert wurden.

- Bei CICS-Anwendungen müssen Sie den Befehl `EXEC CICS SYNCPOINT ROLLBACK` verwenden, um die Arbeitseinheit zurückzusetzen. Verwenden Sie den MQBACK-Aufruf nicht bei CICS-Anwendungen.
- Bei IMS-Anwendungen (außer Stapel-DL/I-Programme) müssen Sie IMS-Aufrufe wie `ROLB` verwenden, um die Arbeitseinheit zurückzusetzen. Verwenden Sie den MQBACK-Aufruf nicht bei IMS-Anwendungen (ausgenommen Stapel-DL/I-Programme).

- Unter IBM i verwenden Sie diesen Aufruf für lokale Arbeitseinheiten, die vom Warteschlangenmanager koordiniert werden. Dies bedeutet, dass auf Jobebene keine COMMIT-Definition existieren darf und somit der Befehl `STRCMTCTL` mit dem Parameter `CMTSCOPE(*JOB)` für den Job nicht ausgegeben worden sein darf.

3. Wird eine Anwendung mit nicht festgeschriebenen Änderungen in einer Arbeitseinheit beendet, ist die Verwerfung dieser Änderungen davon abhängig, ob die Anwendung auf normale oder nicht normale Weise beendet wird. Weitere Informationen finden Sie in den Hinweisen zur Verwendung im Abschnitt „MQDISC - Verbindung mit Warteschlangenmanager beenden" auf Seite 678.
4. Wenn eine Anwendung Nachrichten in Gruppen oder Segmenten logischer Nachrichten einreiht oder abruft, speichert der Warteschlangenmanager die Informationen über die Nachrichtengruppe und logische Nachricht für die letzten erfolgreichen MQPUT- und MQGET-Aufrufe. Diese Informationen sind mit der Warteschlangenkennung verknüpft und umfassen Folgendes:

   - Die Werte der Felder *GroupId*, *MsgSeqNumber*, *Offset* und *MsgFlags* in MQMD
   - Ist die Nachricht Teil einer Arbeitseinheit
   - Bei einem MQPUT-Aufruf: ob die Nachricht persistent oder nicht persistent ist.

   Der Warteschlangenmanager bewahrt *drei* Sätze von Gruppen- und Segmentinformationen zum:

   - Den letzten erfolgreichen MQPUT-Aufruf (dieser kann Teil einer Arbeitseinheit sein).
   - Den letzten erfolgreichen MQGET-Aufruf, durch den eine Nachricht aus der Warteschlange entfernt wurde (dieser kann Teil einer Arbeitseinheit sein).
   - Den letzten erfolgreichen MQGET-Aufruf, mit dem eine Nachricht in der Warteschlange angezeigt wurde (kann *nicht* Teil einer Arbeitseinheit sein).

5. Die mit dem MQGET-Aufruf verknüpften Informationen werden auf den Wert zurückgesetzt, den sie vor dem ersten erfolgreichen MQGET-Aufruf für die betreffende Warteschlangenkennung in der aktuellen Arbeitseinheit hatten.

   Gruppen- und segmentbezogene Informationen von Warteschlangen, die von der Anwendung aktualisiert wurden, nachdem die Arbeitseinheit gestartet wurde, aber außerhalb von deren Bereich, werden nicht wiederhergestellt, wenn die Arbeitseinheit zurückgesetzt wird.

   Werden die vorherigen Werte der gruppen- und segmentbezogenen Informationen wiederhergestellt, wenn eine Arbeitseinheit zurückgesetzt wird, kann die Anwendung eine große Nachrichtengruppe oder eine große logische Nachricht, die aus zahlreichen Segmenten besteht, auf mehrere Arbeitseinheiten verteilen und an der richtigen Stelle in der Nachrichtengruppe oder logischen Nachricht einen Neustart durchführen, wenn eine der Arbeitseinheiten ausfällt.

   Das Verwenden mehrerer Arbeitseinheiten kann von Vorteil sein, wenn der lokale Warteschlangenmanager lediglich über einen geringen Warteschlangenspeicherplatz verfügt. Allerdings muss die Anwendung ausreichend Informationen zur Verfügung haben, um bei einem Systemausfall das Einreihen oder Abrufen von Nachrichten an der richtigen Stelle neu zu starten.

   Weitere Informationen zum Neustart an der korrekten Position nach einem Systemausfall finden Sie unter der Option MQPMO_LOGICAL_ORDER (beschrieben unter „MQPMO - Nachrichteneinreihungsoptionen" auf Seite 484) und der Option MQGMO_LOGICAL_ORDER (beschrieben unter „MQGMO – Nachrichtenabrufoptionen" auf Seite 348).

   Die weiteren Hinweise zur Verwendung sind nur anwendbar, wenn der Warteschlangenmanager die Arbeitseinheiten koordiniert.

6. Eine Arbeitseinheit hat denselben Bereich wie eine Verbindungskennung. Alle MQ-Aufrufe, die eine bestimmte Arbeitseinheit betreffen, müssen mit derselben Verbindungskennung ausgeführt werden. Mit einer anderen Verbindungskennung ausgegebene Aufrufe (zum Beispiel Aufrufe durch eine andere Anwendung) betreffen eine andere Arbeitseinheit. Informationen zum Geltungsbereich von Verbindungskennungen finden Sie in der Beschreibung des Parameters *Hconn* in „MQCONN - Warteschlangenmanager verbinden" auf Seite 655.
7. Dieser Aufruf wirkt sich nur auf Nachrichten aus, die als Teil der aktuellen Arbeitseinheit eingereiht oder abgerufen wurden.
8. Eine Anwendung mit langer Laufzeit, die einen MQGET-, MQPUT- oder MQPUT1-Aufruf in einer Arbeitseinheit ausführt, aber nie eine Commitfunktion oder einen Aufruf zum Zurücksetzen ausführt, kann Warteschlangen mit Nachrichten füllen, die für andere Anwendungen nicht verfügbar sind. Um sich vor dieser Möglichkeit zu schützen, muss der Administrator das Warteschlangenmanagerattribut *MaxUncommittedMsgs* auf einen Wert festlegen, der niedrig genug ist, um ein Auffüllen der Warteschlangen durch solche Anwendungen zu verhindern, aber hoch genug, damit die erwarteten Messaging-Anwendungen ordnungsgemäß arbeiten können.

C-Aufruf

```
MQBACK (Hconn, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Aufruf von High Level Assembler

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Aufruf in Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQBEGIN - Arbeitseinheit starten

Der MQBEGIN-Aufruf startet eine Arbeitseinheit, die vom Warteschlangenmanager koordiniert wird und externe Ressourcenmanager einbeziehen kann.

Syntax

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Hconn muss eine nicht gemeinsam genutzte Verbindungskennung sein. Wenn eine gemeinsam genutzte Verbindungskennung angegeben wird, schlägt der Aufruf mit Ursachencode MQRC_HCONN_ERROR fehl. Weitere Informationen zu gemeinsam genutzten und nicht gemeinsam genutzten Kennungen finden Sie in der Beschreibung der Optionen MQCNO_HANDLE_SHARE_* unter „MQCNO - Verbindungsoptionen” auf Seite 300.

BeginOptions
Typ: MQBO - Ein-/Ausgabe

Hierbei handelt es sich um Optionen, die die Aktion von MQBEGIN steuern, wie unter „MQBO - Startoptionen” auf Seite 260 beschrieben.

Wenn keine Optionen erforderlich sind, können in C oder S/390 Assembler geschriebene Programme eine Nullparameter-Adresse statt der Adresse einer MQBO-Struktur angeben.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_NO_EXTERNAL_PARTICIPANTS
(2121, X'849') Keine teilnehmenden Ressourcenmanager registriert

MQRC_PARTICIPANT_NOT_AVAILABLE
(2122, X'84A') Teilnehmender Ressourcenmanager nicht verfügbar

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_BO_ERROR
(2134, X'856') BeginOptions-Struktur ungültig

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Aufruf in Umgebung nicht gültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UOW_IN_PROGRESS
(2128, X'850') Arbeitseinheit bereits gestartet

Weitere Informationen zu diesen Ursachencodes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Verwenden Sie den MQBEGIN-Aufruf, um eine Arbeitseinheit zu starten, die vom Warteschlangenmanager koordiniert wird und möglicherweise Änderungen an Ressourcen anderer Ressourcenmanager bewirkt. Der Warteschlangenmanager unterstützt drei Typen von Arbeitseinheiten:
   - **Vom Warteschlangenmanager koordinierte lokale Arbeitseinheit**: Eine Arbeitseinheit, für die der Warteschlangenmanager der einzige teilnehmende Ressourcenmanager ist und daher als Arbeitseinheitenkoordinator agiert.
     – Um diesen Typ Arbeitseinheit zu starten, geben Sie die Option MQPMO_SYNCPOINT oder MQGMO_SYNCPOINT beim ersten MQPUT-, MQPUT1- oder MQGET-Aufruf in der Arbeitseinheit an.

- Um diesen Typ Arbeitseinheit festzuschreiben oder zurückzusetzen, verwenden Sie den MQCMIT- oder MQBACK-Aufruf.

- **Vom Warteschlangenmanager koordinierte globale Arbeitseinheit**: Eine Arbeitseinheit, für die der Warteschlangenmanager als der Arbeitseinheitenkoordinator für MQ-Ressourcen *und* für Ressourcen anderer Warteschlangenmanager agiert. Diese Ressourcenmanager arbeiten mit dem Warteschlangenmanager zusammen um sicherzustellen, dass alle Änderungen an Ressourcen in der Arbeitseinheit gemeinsam festgeschrieben oder zurückgesetzt werden.
  - Um diesen Typ Arbeitseinheit zu starten, verwenden Sie den MQBEGIN-Aufruf.
  - Um diesen Typ Arbeitseinheit festzuschreiben oder zurückzusetzen, verwenden Sie den MQCMIT- oder MQBACK-Aufruf.
- **Extern koordinierte globale Arbeitseinheit**: Eine Arbeitseinheit, in der der Warteschlangenmanager ein Teilnehmer ist, aber nicht als Arbeitseinheitenkoordinator agiert. Stattdessen gibt es einen externen Arbeitseinheitenkoordinator, mit dem der Warteschlangenmanager zusammenarbeitet.
  - Um diesen Typ Arbeitseinheit zu starten, verwenden Sie den relevanten Aufruf, der vom externen Arbeitseinheitenkoordinator bereitgestellt wird.

    Wenn der MQBEGIN-Aufruf verwendet wird, um zu versuchen, die Arbeitseinheit zu starten, schlägt der Aufruf mit Ursachencode MQRC_ENVIRONMENT_ERROR fehl.
  - Um diesen Typ Arbeitseinheit festzuschreiben oder zurückzusetzen, verwenden Sie die Festschreibungs- und Rücksetzungsaufrufe, die vom externen Arbeitseinheitenkoordinator bereitgestellt werden.

    Wenn Sie den MQCMIT- oder MQBACK-Aufruf verwenden, um die Arbeitseinheit festzuschreiben oder zurückzusetzen, schlägt der Aufruf mit Ursachencode MQRC_ENVIRONMENT_ERROR fehl.

2. Wenn die Anwendung mit nicht festgeschriebenen Änderungen in einer Arbeitseinheit beendet wird, ist die Disposition dieser Änderungen davon abhängig, ob die Anwendung normal oder abnormal beendet wird. Weitere Informationen finden Sie in den Hinweisen zur Verwendung im Abschnitt „MQDISC - Verbindung mit Warteschlangenmanager beenden" auf Seite 678.
3. Eine Anwendung kann immer jeweils nur an einer Arbeitseinheit teilnehmen. Der MQBEGIN-Aufruf schlägt mit Ursachencode MQRC_UOW_IN_PROGRESS fehl, wenn für die Anwendung bereits eine Arbeitseinheit existiert, unabhängig davon, um welchen Typ Arbeitseinheit es sich handelt.
4. Der MQBEGIN-Aufruf ist in einer MQ MQI-Clientumgebung nicht gültig. Der Versuch, den Aufruf zu verwenden, schlägt mit Ursachencode MQRC_ENVIRONMENT_ERROR fehl.
5. Wenn der Warteschlangenmanager als Arbeitseinheitenkoordinator bei globalen Arbeitseinheiten agiert, werden die Ressourcenmanager, die an der Arbeitseinheit teilnehmen können, in der Konfigurationsdatei des Warteschlangenmanagers definiert.
6. Unter IBM i werden die drei Typen von Arbeitseinheiten folgendermaßen unterstützt:
   - **Die vom Warteschlangenmanager koordinierte lokale Arbeitseinheit** kann nur verwendet werden, wenn auf der Jobebene keine COMMIT-Definition existiert, d. h., der Befehl STRCMTCTL mit dem Parameter `CMTSCOPE(*JOB)` darf nicht für den Job ausgegeben worden sein.
   - **Die vom Warteschlangenmanager koordinierte globale Arbeitseinheit** wird nicht unterstützt.
   - **Die extern koordinierte globale Arbeitseinheit** kann nur verwendet werden, wenn eine COMMIT-Definition auf Jobebene existiert, d. h., der Befehl STRCMTCTL mit dem Parameter `CMTSCOPE(*JOB)` muss für den Job ausgegeben worden sein. Wenn dies zutrifft, gelten die IBM i `COMMIT`- und `ROLLBACK`-Operationen für MQ-Ressourcen sowie für Ressourcen, die anderen teilnehmenden Ressourcenmanagern gehören.

C-Aufruf

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;         /* Connection handle */
MQBO     BeginOptions;  /* Options that control the action of MQBEGIN */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
     CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Konvertieren von Puffern in Nachrichtenkennungen

Der Funktionsaufruf MQBUFMH konvertiert einen Puffer in eine Nachrichtenkennung und ist die Umkehrfunktion des Aufrufs MQMHBUF.

Dieser Aufruf erfasst einen Nachrichtendeskriptor und MQRFH2-Eigenschaften im Puffer und macht sie über eine Nachrichtenkennung verfügbar. Die MQRFH2-Eigenschaften in den Nachrichtendaten werden optional entfernt. Die Felder *Encoding*, *CodedCharSetId*und *Format* des Nachrichtendeskriptors werden bei Bedarf aktualisiert, um den Inhalt des Puffers nach dem Entfernen der Eigenschaften korrekt zu beschreiben.

Syntax

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *Buffer*, *BufferLength*, *DataLength*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* muss der Verbindungskennung entsprechen, die zum Erstellen der im Parameter *Hmsg* angegebenen Nachrichtenkennung verwendet wurde.

Wenn die Nachrichtenkennung mit MQHC_UNASSOCIATED_HCONN erstellt wurde, muss eine gültige Verbindung in dem Thread erstellt werden, der einen Puffer in eine Nachrichtenkennung konvertiert. Wird keine gültige Verbindung hergestellt, schlägt der Aufruf mit MQRC_CONNECTION_BROKEN fehl.

Hmsg
Typ: MQHMQSG - Eingabe

Dies ist die Nachrichtenkennung, für die ein Puffer erforderlich ist. Der Wert wurde durch einen früheren MQCRTMH-Aufruf zurückgemeldet.

BufMsgHOpts
Typ: MQBMHO - Eingabe

Über die MQBMHO-Struktur können Anwendungen Optionen zur Steuerung der Erstellung von Nachrichtenkennungen aus Puffern festlegen.

Details siehe „MQBMHO - Puffer-zu-Nachrichtenhandle-Optionen” auf Seite 258.

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Die Struktur *MsgDesc* enthält die Nachrichtendeskriptoreigenschaften und beschreibt den Inhalt des Pufferbereichs.

Bei Ausgabe des Aufrufs werden die Eigenschaften optional aus dem Pufferbereich entfernt und, in diesem Fall, der Nachrichtendeskriptor wird so aktualisiert, dass der Pufferbereich korrekt beschrieben wird.

Die Daten in dieser Struktur müssen im Zeichensatz und in der Codierung der Anwendung vorliegen.

BufferLength
Typ: MQLONG - Eingabe

BufferLength ist die Länge des Pufferbereichs in Byte.

Eine *BufferLength* von null Byte ist gültig und gibt an, dass der Pufferbereich keine Daten enthält.

Buffer
Typ: MQBYTExBufferLength - Ein-/Ausgabe

Hierbei handelt es sich um Optionen, die die Aktion von MQBEGIN steuern, wie unter „MQBEGIN - Arbeitseinheit starten” auf Seite 623 beschrieben.

Buffer definiert den Bereich, der den Nachrichtenpuffer enthält. Für die meisten Daten sollten Sie den Puffer an einem 4-Byte-Grenzwert ausrichten.

Wenn *Buffer* Zeichendaten oder numerische Daten enthält, setzen Sie die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* auf die für die Daten geeigneten Werte. Dadurch können die Daten bei Bedarf konvertiert werden.

Befinden sich Eigenschaften im Nachrichtenpuffer, werden sie optional entfernt; bei Rückgabe des Aufrufs sind sie später über die Nachrichtenkennung wieder verfügbar.

In der Programmiersprache C ist der Parameter als ein Zeiger-auf-typenlos deklariert, d. h., dass die Adresse eines beliebigen Datentyps als Parameter angegeben werden kann.

Wenn der Parameter *BufferLength* null ist, wird *Buffer* nicht referenziert. In diesem Fall kann die Parameteradresse, die von Programmen übergeben wird, die in C oder System/390 Assembler geschrieben sind, null sein.

DataLength
Typ: MQLONG - Ausgabe

Die Länge des Puffers, in dem möglicherweise Eigenschaften entfernt wurden, in Bytes.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BMHO_ERROR
(2489, X'09B9') Struktur der Puffer-zu-Nachrichtenkennung-Optionen nicht gültig.

MQRC_BUFFER_ERROR
(2004, X'07D4') Pufferparameter nicht gültig.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Pufferlängenparameter nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennung nicht gültig.

MQRC_MD_ERROR
(2026, X'07EA') Nachrichtendeskriptor nicht gültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_RFH_ERROR
(2334, X'091E') MQRFH2-Struktur nicht gültig.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975') Ein MQRFH2-Ordner, der Eigenschaften enthält, konnte nicht analysiert werden.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

MQBUFMH-Aufrufe können nicht von API-Exits abgefangen werden – ein Puffer wird im Anwendungsspeicher in eine Nachrichtenkennung konvertiert. Der Aufruf erreicht den Warteschlangenmanager nicht.

C-Aufruf

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                           BUFFER, DATALENGTH, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
              DataLength, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQBUFMH,(HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
               DATALENGTH,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 – Callback verwalten

Der MQCB-Aufruf registriert einen Callback für die angegebene Objektkennung und steuert die Aktivierung des Callback und der an ihm vorgenommenen Änderungen.

Ein Callback ist ein Code-Stück (angegeben entweder als Name einer dynamisch verknüpfbaren Funktion oder als Funktionszeiger), das beim Auftreten bestimmter Ereignisse von IBM WebSphere MQ aufgerufen wird.

Um MQCB und MQCTL auf einem V7-Client zu verwenden, müssen Sie mit einem V7-Server verbunden sein und der Parameter **SHARECNV** des Kanals muss einen Wert ungleich null haben.

Folgende Callback-Typen können definiert werden:

Nachrichtenkonsument
Eine Callback-Funktion für einen Nachrichtenkonsumenten wird aufgerufen, wenn eine Nachricht, die die angegebenen Auswahlkriterien erfüllt, an einer Objektkennung verfügbar ist.

Für jede Objektkennung kann nur eine Callback-Funktion registriert werden. Soll nur eine Warteschlange mit mehreren Auswahlkriterien gelesen werden, muss die Warteschlange mehrere Male geöffnet und für jede Kennung eine Konsumentenfunktion registriert werden.

Event handler (Ereigniskennung)
Der Ereignishandler wird bei Bedingungen aufgerufen, die sich auf die gesamte Callback-Umgebung auswirken.

Die Funktion wird bei Eintreten einer Ereignisbedingung aufgerufen, etwa wenn der Warteschlangenmanager oder die Verbindung beendet wird oder in den Quiescemodus wechselt.

Die Funktion wird nicht für Bedingungen aufgerufen, die für einen einzelnen Nachrichtenkonsumenten gelten, wie zum Beispiel MQRC_GET_INHIBITED; sie wird hingegen aufgerufen, wenn eine Callback-Funktion nicht normal beendet wird.

Syntax

MQCB (*Hconn*, *Operation*, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

In z/OS für CICS -Anwendungen und in IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, können Sie den folgenden Sonderwert für *MQHC_DEF_HCONN* angeben, um die dieser Ausführungseinheit zugeordnete Verbindungskennung zu verwenden.

Operation

Typ: MQLONG - Eingabe

Die Operation, die für den Callback verarbeitet wird, die für die angegebene Objektkennung definiert ist. Sie müssen eine der folgenden Optionen angeben; wenn mehrere Optionen erforderlich sind, sind folgende Werte möglich:

- Hinzufügen (fügen Sie dieselbe Konstante nicht mehrmals hinzu) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

MQOP_REGISTER

Definiert die Callback-Funktion für die angegebene Objektkennung. Diese Operation legt fest, welche Funktion aufgerufen wird und welche Auswahlkriterien verwendet werden sollen.

Ist bereits eine Callback-Funktion für die Objektkennung definiert, wird die Definition ersetzt. Wird beim Ersetzen des Callback ein Fehler erkannt, wird die Registrierung der Funktion aufgehoben.

Wenn eine Callback-Funktion in derselben Callback-Funktion registriert wird, deren Registrierung zuvor aufgehoben wurde, wird dies als Austauschoperation behandelt; die ursprünglichen und endgültigen Aufrufe werden nicht getätigt.

Sie können MQOP_REGISTER in Verbindung mit MQOP_SUSPEND oder MQOP_RESUME verwenden.

MQOP_DEREGISTER

Beendet die Verarbeitung von Nachrichten für die Objektkennung und entfernt die Kennung aus für einen Callback infrage kommenden Nachrichten.

Die Registrierung eines Callback wird automatisch aufgehoben, wenn die zugehörige Kennung geschlossen wird.

Wenn MQOP_DEREGISTER aus einem Nutzer aufgerufen wird und für den Callback ein Anrufstopp definiert wurde, wird es bei der Rückgabe vom Nutzer aufgerufen.

Wenn diese Operation für einen *Hobj* ohne registrierten Konsumenten ausgegeben wird, wird der Aufruf mit MQRC_CALLBACK_NOT_REGISTERED zurückgegeben.

MQOP_SUSPEND

Setzt die Verarbeitung von Nachrichten für die Objektkennung aus.

Wird diese Operation auf einen Ereignishandler angewendet, ruft er im ausgesetzten Zustand keine Ereignisse ab. Ereignisse, die in diesem Zustand nicht erfasst wurden, werden der Operation nicht bereitgestellt, wenn sie fortgesetzt wird.

Im ausgesetzten Zustand ruft die Konsumentenfunktion weiterhin Callbacks des Steuerungstyps ab.

MQOP_RESUME

Setzt die Verarbeitung von Nachrichten für die Objektkennung fort.

Wird diese Operation auf einen Ereignishandler angewendet, ruft er im ausgesetzten Zustand keine Ereignisse ab. Ereignisse, die in diesem Zustand nicht erfasst wurden, werden der Operation nicht bereitgestellt, wenn sie fortgesetzt wird.

CallbackDesc
Typ: MQCBD - Eingabe

Dies ist eine Struktur, die die Callback-Funktion identifiziert, die von der Anwendung registriert wird, sowie die für die Registrierung verwendeten Optionen.

Weitere Informationen zu dieser Struktur finden Sie unter MQCBD.

Ein Callback-Deskriptor wird nur für die Option MQOP_REGISTER benötigt. Wenn der Deskriptor nicht benötigt wird, kann die übergebene Parameteradresse null sein.

Hobj
Typ: MQHOBJ - Eingabe

Diese Kennung steht für den Zugriff, der für das Objekt eingerichtet wurde, von dem eine Nachricht verarbeitet werden soll. Dies ist eine Kennung, die von einem vorherigen MQOPEN -oder MQSUB -Aufruf (im Parameter *Hobj*) zurückgegeben wurde.

Hobj wird für die Definition einer Routine für die Ereigniskennung (MQCBT_EVENT_HANDLER) nicht benötigt und sollte als MQHO_NONE angegeben werden.

Wenn *Hobj* von einem MQOPEN-Aufruf zurückgemeldet wurde, muss die Option mit mindestens einer der folgenden Optionen geöffnet worden sein:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc
Typ: MQMD - Eingabe

Diese Struktur beschreibt die Attribute der erforderlichen Nachricht und die der abgerufenen Nachricht.

Der Parameter *MsgDesc* definiert die vom Konsumenten benötigten Attribute der Nachricht sowie die Version des an den Nachrichtenkonsumenten übergebenen MQMD.

Über die Optionen *MsgId*, *CorrelId*, *GroupId* , *MsgSeqNumber* und *Offset* in der MQMD werden in Abhängigkeit von den im Parameter *GetMsgOpts* angegebenen Optionen Nachrichten ausgewählt.

Encoding und *CodedCharSetId* werden zur Nachrichtenkonvertierung verwendet, wenn Sie die Option MQGMO_CONVERT angeben.

Details hierzu finden Sie im Abschnitt MQMD.

MsgDesc wird für MQOP_REGISTER verwendet und wenn Sie andere Werte als die Standardwerte für irgendwelche Felder benötigen. *MsgDesc* wird nicht für einen Ereignishandler verwendet.

Wenn der Deskriptor nicht benötigt wird, kann die übergebene Parameteradresse null sein.

Sind mehrere Konsumenten bei derselben Warteschlange mit einander überschneidender Auswahl registriert, ist der für jede Nachricht ausgewählte Konsument nicht definiert.

GetMsgOpts
Typ: MQGMO - Eingabe

Der Parameter *GetMsgOpts* steuert, wie der Nachrichtenkonsument Nachrichten abruft. Alle Optionen dieses Parameters haben die in „MQGMO – Nachrichtenabrufoptionen” auf Seite 348 beschriebene Bedeutung, wenn sie in einem MQGET-Aufruf verwendet werden, außer:

MQGMO_SET_SIGNAL
Diese Option ist nicht zulässig.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_*

Die Reihenfolge, in der Nachrichten beim Browsen an einen Konsumenten geliefert werden, wird durch Kombination dieser Optionen bestimmt. Wichtige Kombinationen sind:

MQGMO_BROWSE_FIRST

Die erste Nachricht in der Warteschlange wird wiederholt an den Konsumenten übermittelt. Das ist praktisch, wenn der Konsument die Nachricht bei seinem Callback zerstört. Verwenden Sie diese Option mit Vorsicht.

MQGMO_BROWSE_NEXT

Der Konsument erhält jede Nachricht aus der Warteschlange, von der aktuellen Cursorposition bis zum Ende der Warteschlange.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT

Der Cursor wird an den Anfang der Warteschlange zurückgesetzt. Der Konsument erhält danach jede Nachricht, bis der Cursor das Ende der Warteschlange erreicht.

MQGMO_BROWSE_FIRST + MQGMO_MARK_*

Beginnend am Anfang der Warteschlange erhält der Konsument die erste nicht markierte Nachricht in der Warteschlange, die danach für diesen Konsumenten markiert wird. Diese Kombination stellt sicher, dass der Konsument neue Nachrichten empfangen kann, die nach der aktuellen Cursorposition hinzugefügt werden.

MQGMO_BROWSE_NEXT + MQGMO_MARK_*

Beginnend an der Cursorposition erhält der Konsument die nächste nicht markierte Nachricht in der Warteschlange, die danach für diesen Konsumenten markiert wird. Verwenden Sie diese Kombination mit Vorsicht, da Nachrichten hinter der aktuellen Cursorposition zur Warteschlange hinzugefügt werden können.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_*

Diese Kombination ist nicht zulässig. Wird sie verwendet, meldet der Aufruf MQRC_OPTIONS_ERROR zurück.

MQGMO_NO_WAIT, MQGMO_WAIT und WaitInterval

Diese Optionen steuern, wie der Verbraucher aufgerufen wird.

MQGMO_NO_WAIT

Der Konsument wird niemals mit MQRC_NO_MSG_AVAILABLE aufgerufen. Der Konsument wird nur bei Nachrichten und Ereignissen aufgerufen.

MQGMO_WAIT mit WaitInterval Null

Der Code MQRC_NO_MSG_AVAILABLE wird an den Konsumenten übergeben, wenn keine Nachrichten vorhanden sind und entweder der Konsument gestartet wurde oder der Konsument seit dem letzten Ursachencode mindestens eine Nachricht erhalten hat.

Auf diese Weise wird verhindert, dass der Konsument in einer ausgelasteten Schleife eine Abfrage durchführt, wenn ein Warteintervall mit dem Wert null angegeben ist.

MQGMO_WAIT und positives WaitInterval

Der Konsument wird nach dem angegebenen Warteintervall mit dem Ursachencode MQRC_NO_MSG_AVAILABLE aufgerufen. Dieser Aufruf wird unabhängig davon durchgeführt, ob dem Konsumenten Nachrichten übermittelt wurden. Auf diese Weise kann der Benutzer eine Heartbeat- oder Stapelverarbeitung durchführen.

MQGMO_WAIT und WaitInterval von MQWI_UNLIMITED

Gibt eine unendliche Wartezeit an, bis MQRC_NO_MSG_AVAILABLE zurückgemeldet wird. Der Konsument wird niemals mit MQRC_NO_MSG_AVAILABLE aufgerufen.

GetMsgOpts wird nur für MQOP_REGISTER verwendet und wenn Sie für irgendwelche Felder andere Werte als die Standardwerte benötigen. *GetMsgOpts* wird nicht für einen Ereignishandler verwendet.

Wenn die *GetMsgOpts* nicht erforderlich sind, kann die übergebene Parameteradresse null sein. Dieser Parameter hat dieselbe Wirkung wie die Angabe von MQGMO_DEFAULT zusammen mit MQGMO_FAIL_IF_QUIESCING.

Wenn die Kennung einer Nachrichteneigenschaft in der MQGMO-Struktur angegeben wird, wird in der MQGMO-Struktur eine Kopie angefertigt, die in den Callback an den Konsumenten übergeben wird. Bei Rückgabe vom MQCB-Aufruf kann die Anwendung die Kennung der Nachrichteneigenschaft löschen.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Die Ursachencodes in der folgenden Liste sind diejenigen, die der Warteschlangenmanager für den Parameter *Reason* zurückgeben kann.

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') Servicemodule für Datenkonvertierung können nicht geladen werden.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Falsches Rückruftypfeld.

MQRC_CALLBACK_NOT_REGISTERED
(2448, X'990') Aufhebung der Registrierung, Aussetzen oder Fortsetzen nicht möglich, weil kein Rückruf registriert wurde.

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Entweder *CallbackFunction* oder *CallbackName* muss angegeben sein, aber nicht beides.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Falsches Rückruftypfeld.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Falsches Feld für MQCBD-Optionen.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Fehler bei Korrelations-ID.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Parameter Datenlänge ungültig.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') Die angeforderte Funktion ist in der aktuellen Umgebung nicht verfügbar.

MQRC_GET_INHIBITED
(2016, X'7E0') wird für die Warteschlange unterdrückt.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Konflikt mit globaler Arbeitseinheit.

MQRC_GMO_ERROR
(2186, X'88A') Optionsstruktur zum Nachrichtenabruf ungültig.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Kennung für globale Arbeitseinheit belegt.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') Spezifikation zum Durchsuchen inkonsistent.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Nachricht unter Cursor nicht gültig für Abruf

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') Globale Arbeitseinheit im Konflikt mit lokaler Arbeitseinheit.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Abgleichoptionen ungültig

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B4') Falsches Feld *MaxMsgLength* .

MQRC_MD_ERROR
(2026, X'7EA') Nachrichtendeskriptor ungültig

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') Der Funktionseingangspunkt konnte im Modul nicht gefunden werden.

MQRC_MODULE_INVALID
(2496, X'9C0') Modul gefunden, jedoch ist der Typ falsch; weder 32 Bit noch 64 Bit, noch eine gültige Dynamic Link Library.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Modul im Suchpfad nicht gefunden oder keine Berechtigung zum Laden.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Nachrichtenfolgenummer ungültig

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Verwendung des Nachrichtentokens ungültig.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Keine Nachricht verfügbar.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') Anzeigecursor nicht auf Nachricht positioniert.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Warteschlange nicht für Anzeige geöffnet

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Warteschlange nicht zur Eingabe geöffnet.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OPERATION_ERROR
(2206, X'89E') Operationscode für API-Aufruf falsch.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Warteschlange hat falschen Indextyp

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Signal für diese Kennung ausstehend.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') In der aktuellen Arbeitseinheit können keine weiteren Nachrichten verarbeitet werden.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') Unterstützung für Synchronisationspunkt nicht verfügbar.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Eintragung in globale Arbeitseinheit fehlgeschlagen.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') Mischung von Aufrufen für Arbeitseinheiten wird nicht unterstützt.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') Arbeitseinheit nicht zur Verwendung durch den Warteschlangenmanager verfügbar.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') Warteintervall in MQGMO ungültig

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Falsche MQGMO-Version bereitgestellt.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Falsche MQMD-Version bereitgestellt.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Mit MQCB wird die für jede Nachricht in der Warteschlange aufzurufende Aktion unter Berücksichtigung der angegebenen Kriterien definiert. Bei der Verarbeitung der Aktion wird entweder die Nachricht aus der Warteschlange entfernt und zum vorgegebenen Nachrichtenkonsumenten übermittelt oder es wird ein Nachrichtentoken zum Abrufen der Nachricht bereitgestellt.
2. Mit MQCB können Callback-Routinen definiert werden, bevor die Verarbeitung mit MQCTL gestartet wird, oder die Option kann innerhalb einer Callback-Routine eingesetzt werden.
3. Um MQCB außerhalb einer Callback-Routine zu verwenden, müssen Sie zunächst über MQCTL die Nachrichtenverarbeitung aussetzen und anschließend fortsetzen.
4. MQCB wird im IMS -Adapter nicht unterstützt.

Nachrichtenkonsumenten-Callbacksequenz

Sie können einen Konsumenten so konfigurieren, dass ein Callback an wichtigen Punkten im Lebenszyklus des Konsumenten aufgerufen wird. Beispiel:

- erste Registrierung des Konsumenten
- Herstellung der Verbindung
- Unterbrechung der Verbindung
- Aufheben der Registrierung des Konsumenten, entweder explizit oder implizit durch ein MQCLOSE

Tabelle 562. MQCTL-Verbdefinitionen

Verb	**Bedeutet**
MQCTL(START)	MQCTL-Aufruf über die Operation MQOP_START
MQCTL(STOP)	MQCTL-Aufruf über die Operation MQOP_STOP
MQCTL(WAIT)	MQCTL-Aufruf über die Operation MQOP_START_WAIT

So kann der Konsument den Status beibehalten, der ihm zugeordnet wurde. Wird ein Callback von einer Anwendung angefordert, gelten folgende Regeln für den Konsumentenaufruf:

REGISTER
Ist immer der erste Aufruftyp des Callbacks.

Wird immer für denselben Thread aufgerufen wie der Aufruf MQCB(REGISTER).

START
Wird immer synchron mit dem Verb MQCTL(START) aufgerufen.

- Alle START-Callbacks werden ausgeführt, bevor das Verb MQCTL(START) zurückgegeben wird.

  Befindet sich im gleichen Thread wie die Nachrichtenübermittlung, wenn THREAD_AFFINITY angefordert wird.

  Der Aufruf mit Start ist nicht garantiert, wenn beispielsweise ein vorheriger Callback MQCTL(STOP) während des MQCTL(START) ausgibt.

STOPP
Nachrichten oder Ereignisse werden nach diesem Aufruf erst wieder übermittelt, nachdem die Verbindung wiederhergestellt wurde.

Ein STOP ist garantiert, wenn die Anwendung zuvor für START oder für eine Nachricht oder ein Ereignis aufgerufen wurde.

DEREGISTER
Ist immer der letzte Aufruftyp des Callbacks.

Stellen Sie sicher, dass Ihre Anwendung in den START- und STOP-Callbacks eine Thread-basierte Initialisierung und Bereinigung durchführt. Eine nicht Thread-basierte Initialisierung und Bereinigung können Sie mit den Callbacks REGISTER und DEREGISTER ausführen.

Stellen Sie keine Vermutungen über die Lebensdauer und Verfügbarkeit des Threads an, außer den angegebenen. Verlassen Sie sich z. B. nicht darauf, dass ein Thread über den letzten Aufruf DEREGISTER hinaus aktiv bleibt. Ebenso dürfen Sie, wenn Sie THREAD_AFFINITY nicht verwenden möchten, nicht davon ausgehen, dass der Thread bei jedem Starten der Verbindung existiert.

Wenn Ihre Anwendung bestimmte Anforderungen an die Eigenschaften von Threads stellt, kann sie immer einen entsprechenden Thread erstellen. Verwenden Sie dann MQCTL(WAIT). Hierdurch wird der Thread zur asynchronen Nachrichtenbereitstellung an IBM WebSphere MQ "gespendet".

Verwendung der Nachrichtenkonsumentenverbindung

Sie können einen Konsumenten so konfigurieren, dass ein Callback an wichtigen Punkten im Lebenszyklus des Konsumenten aufgerufen wird. Beispiel:

- erste Registrierung des Konsumenten
- Herstellung der Verbindung
- Unterbrechung der Verbindung
- Aufheben der Registrierung des Konsumenten, entweder explizit oder implizit durch ein MQCLOSE

Tabelle 563. MQCTL-Verbdefinitionen

Verb	**Bedeutet**
MQCTL(START)	MQCTL-Aufruf über die Operation MQOP_START
MQCTL(STOP)	MQCTL-Aufruf über die Operation MQOP_STOP
MQCTL(WAIT)	MQCTL-Aufruf über die Operation MQOP_START_WAIT

So kann der Konsument den Status beibehalten, der ihm zugeordnet wurde. Wird ein Callback von einer Anwendung angefordert, gelten folgende Regeln für den Konsumentenaufruf:

REGISTER
Ist immer der erste Aufruftyp des Callbacks.

Wird immer für denselben Thread aufgerufen wie der Aufruf MQCB(REGISTER).

START
Wird immer synchron mit dem Verb MQCTL(START) aufgerufen.

- Alle START-Callbacks werden ausgeführt, bevor das Verb MQCTL(START) zurückgegeben wird.

Befindet sich im gleichen Thread wie die Nachrichtenübermittlung, wenn THREAD_AFFINITY angefordert wird.

Der Aufruf mit Start ist nicht garantiert, wenn beispielsweise ein vorheriger Callback MQCTL(STOP) während des MQCTL(START) ausgibt.

STOPP
Nachrichten oder Ereignisse werden nach diesem Aufruf erst wieder übermittelt, nachdem die Verbindung wiederhergestellt wurde.

Ein STOP ist garantiert, wenn die Anwendung zuvor für START oder für eine Nachricht oder ein Ereignis aufgerufen wurde.

DEREGISTER
Ist immer der letzte Aufruftyp des Callbacks.

Stellen Sie sicher, dass Ihre Anwendung in den START- und STOP-Callbacks eine Thread-basierte Initialisierung und Bereinigung durchführt. Eine nicht Thread-basierte Initialisierung und Bereinigung können Sie mit den Callbacks REGISTER und DEREGISTER ausführen.

Stellen Sie keine Vermutungen über die Lebensdauer und Verfügbarkeit des Threads an, außer den angegebenen. Verlassen Sie sich z. B. nicht darauf, dass ein Thread über den letzten Aufruf DEREGISTER hinaus aktiv bleibt. Ebenso dürfen Sie, wenn Sie THREAD_AFFINITY nicht verwenden möchten, nicht davon ausgehen, dass der Thread bei jedem Starten der Verbindung existiert.

Wenn Ihre Anwendung bestimmte Anforderungen an die Eigenschaften von Threads stellt, kann sie immer einen entsprechenden Thread erstellen. Verwenden Sie dann MQCTL(WAIT). Hierdurch wird der Thread zur asynchronen Nachrichtenbereitstellung an IBM WebSphere MQ "gespendet".

C-Aufruf

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
GetMsgOpts, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,
            GETMSGOPTS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
          CompCode, Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Callback-Funktion

Der Funktionsaufruf MQCB_FUNCTION ist die Callback-Funktion für die Ereignisverarbeitung und Verarbeitung von asynchronen Nachrichten.

Die Aufrufdefinition MQCB_FUNCTION wird lediglich zur Beschreibung der an die Callback-Funktion übergebenen Parameter bereitgestellt. Der Warteschlangenmanager stellt keinen Einstiegspunkt namens MQCB_FUNCTION bereit.

Die Spezifikation der eigentlichen aufzurufenden Funktion ist eine Eingabe für den MQCB-Aufruf, die über die MQCBD-Struktur übergeben wird.

Syntax

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben. Bei z/OS für CICS-Anwendungen und IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und folgender Wert für "Hconn" angegeben werden:

MQHC_DEF_CONN

Standardverbindungskennung

MsgDesc

Typ: MQMD - Eingabe

Diese Struktur beschreibt die Attribute der abgerufenen Nachricht.

Details siehe „MQMD - Nachrichtendeskriptor” auf Seite 399.

Die übergebene MQMD-Version ist dieselbe Version, die im MQCB-Aufruf übergeben wurde, mit dem die Konsumentenfunktion definiert wurde.

Die Adresse des MQMD wird als Nullzeichen übergeben, wenn MQGMO Version 4 verwendet wurde, um anzufordern, dass anstelle eines MQMD eine Nachrichtenkennung zurückgemeldet werden soll.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion; sie ist für eine Ereignishandler-Funktion ohne Relevanz.

GetMsgOpts
Typ: MQGMO - Eingabe

Optionen für die Steuerung der Aktionen des Nachrichtenkonsumenten. Dieser Parameter enthält außerdem zusätzliche Informationen über die zurückgemeldete Nachricht.

Details siehe MQGMO.

Die übergebene MQGMO-Version ist die aktuellste unterstützte Version.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion; sie ist für eine Ereignishandler-Funktion ohne Relevanz.

Buffer
Typ: MQBYTExBufferLength - Eingabe

Dieser Bereich enthält die Nachrichtendaten.

Wenn für diesen Aufruf keine Nachricht verfügbar ist oder wenn die Nachricht keine Nachrichtendaten enthält, wird die Adresse des *Buffer* als Nullen übergeben.

Dies ist ein Eingabefeld für die Nachrichtenkonsumentenfunktion; sie ist für eine Ereignishandler-Funktion ohne Relevanz.

Context
Typ: MQCBC - Ein-/Ausgabe

Diese Struktur stellt Kontextinformationen für die Callback-Funktionen bereit. Details siehe „MQCBC – Callback-Kontext” auf Seite 262.

Hinweise zur Verwendung

1. Denken Sie daran, dass Ihre Callback-Routinen Services verwenden, die den Thread verzögern oder blockieren könnten; zum Beispiel könnte MQGET mit "wait" das Versenden anderer Callbacks verzögern.
2. Es wird nicht automatisch eine separate Arbeitseinheit für jeden Aufruf einer Callback-Routine eingerichtet, also können Routinen entweder einen Commit aufrufen oder ein Commit verzögern, bis ein logischer Arbeitsstapel verarbeitet wurde. Wenn der Arbeitsstapel festgeschrieben wird, werden die Nachrichten für alle Callback-Funktionen festgeschrieben, die seit dem letzten Synchronisationspunkt aufgerufen wurden.
3. Von CICS LINK oder CICS START aufgerufene Programme rufen Parameter mithilfe von CICS-Services über Namensobjekte ab, die als Channel-Container bezeichnet werden. Die Containernamen sind mit den Parameternamen identisch. Weitere Informationen hierzu finden Sie in der CICS-Dokumentation.
4. Callback-Routinen können einen MQDISC-Aufruf ausgeben, aber nicht für ihre eigene Verbindung. Wenn zum Beispiel eine Callback-Routine eine Verbindung hergestellt hat, kann sie diese auch wieder trennen.
5. Eine Callback-Routine sollte grundsätzlich nicht jedes Mal von demselben Thread aufgerufen werden müssen. Falls erforderlich, verwenden Sie MQCTLO_THREAD_AFFINITY, nachdem die Verbindung hergestellt wurde.
6. Wenn eine Callback-Routine einen Ursachencode ungleich null erhält, muss sie angemessene Maßnahmen ergreifen.
7. MQCB_FUNCTION wird im IMS -Adapter nicht unterstützt.

MQCLOSE - Objekt schließen

Der MQCLOSE-Aufruf gibt den Zugriff auf ein Objekt frei und ist die Umkehrfunktion der Aufrufe MQOPEN und MQSUB.

Syntax

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS-Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, können Sie den MQCONN-Aufruf übergehen und folgende Werte für *Hconn* angeben:

MQHC_DEF_HCONN
Standardverbindungskennung

Hobj

Typ: MQHOBJ - Ein-/Ausgabe

Diese Kennung steht für das Objekt, das geschlossen wird. Dabei kann es sich um das Objekt eines beliebigen Typs handeln. Der Wert von *Hobj* wurde von einem vorherigen MQOPEN-Aufruf zurückgegeben.

Bei erfolgreicher Beendigung des Aufrufs setzt der Warteschlangenmanager diesen Parameter auf einen Wert, der keine gültige Kennung für die Umgebung darstellt. Dieser Wert lautet:

MQHO_UNUSABLE_HOBJ
Unbrauchbare Objektkennung

Unter z/OS wird *Hobj* auf einen Wert gesetzt, der nicht definiert ist.

Optionen

Typ: MQLONG - Eingabe

Dieser Parameter steuert, wie das Objekt geschlossen wird.

Nur permanente dynamische Warteschlangen und Subskriptionen können auf mehrere Arten geschlossen werden, weil sie entweder beibehalten oder gelöscht werden müssen. Dabei handelt es sich um Warteschlangen mit dem Attribut *DefinitionType*, das den Wert MQQDT_PERMANENT_DYNAMIC hat (siehe Beschreibung des Attributs *DefinitionType* im Abschnitt „Attribute für Warteschlangen" auf Seite 833). Dieser Abschnitt enthält eine Zusammenfassung der Optionen für das Schließen von Objekten.

Permanente Subskriptionen können entweder beibehalten oder entfernt werden; diese Subskriptionen werden mit dem MQSUB-Aufruf und der Option MQSO_DURABLE erstellt.

Beim Schließen der Kennung für ein verwaltetes Ziel (d. h., der Parameter *Hobj* wurde in einem MQSUB-Aufruf mit der Option MQSO_MANAGED zurückgegeben) bereinigt der Warteschlangenmanager alle nicht abgerufenen Veröffentlichungen, wenn auch die zugehörige Subskription entfernt wurde. Die Subskription wird mit der Option MQCO_REMOVE_SUB für den Parameter *Hsub*, der in einem MQSUB-Aufruf zurückgegeben wird, entfernt. Beachten Sie, dass MQCO_REMOVE_SUB das Standardverhalten von MQCLOSE für eine nicht permanente Subskription darstellt.

Wenn Sie eine Kennung für ein nicht verwaltetes Ziel schließen, müssen Sie selbst die Warteschlange bereinigen, an die Veröffentlichungen gesendet werden. Schließen Sie zunächst die Subskription mit der Option MQCO_REMOVE_SUB und verarbeiten Sie dann Nachrichten aus der Warteschlange, bis diese leer ist.

Sie können nur eine der folgenden Optionen angeben:

Optionen für dynamische Warteschlangen: Diese Optionen steuern, wie permanente dynamische Warteschlangen geschlossen werden.

MQCO_DELETE
Die Warteschlange wird gelöscht, wenn eine der folgenden Bedingungen zutrifft:

- Es handelt sich um eine permanente dynamische Warteschlange, erstellt mit einem vorherigen MQOPEN-Aufruf, es befinden sich keine Nachrichten in der Warteschlange und es stehen keine GET- oder PUT-Anforderungen für die Warteschlange an (weder für die aktuelle Aufgabe noch für irgendeine andere Aufgabe).
- Es handelt sich um eine temporäre dynamische Warteschlange, die von dem MQOPEN-Aufruf erstellt wurde, der *Hobj* zurückgegeben hat. In diesem Fall werden alle Nachrichten in der Warteschlange gelöscht.

In allen anderen Fällen, auch wenn der Parameter *Hobj* mit einem MQSUB-Aufruf zurückgegeben wurde, schlägt der Aufruf mit Ursachencode MQRC_OPTION_NOT_VALID_FOR_TYPE fehl und das Objekt wird nicht gelöscht.

Unter z/OS wird die Warteschlange physisch gelöscht, wenn es sich um eine dynamische Warteschlange handelt, die logisch gelöscht wurde, und dies die letzte Kennung für die Warteschlange ist. Weitere Informationen finden Sie im Abschnitt „Hinweise zur Verwendung” auf Seite 648.

MQCO_DELETE_PURGE
Die Warteschlange und alle darin enthaltenen Nachrichten werden gelöscht, wenn eine der folgenden Bedingungen zutrifft:

- Es handelt sich um eine permanente dynamische Warteschlange, erstellt mit einem vorherigen MQOPEN-Aufruf, und es stehen keine GET- oder PUT-Anforderungen für die Warteschlange an (weder für die aktuelle Aufgabe noch für irgendeine andere Aufgabe).
- Es handelt sich um eine temporäre dynamische Warteschlange, die von dem MQOPEN-Aufruf erstellt wurde, der *Hobj* zurückgegeben hat.

In allen anderen Fällen, auch wenn der Parameter *Hobj* mit einem MQSUB-Aufruf zurückgegeben wurde, schlägt der Aufruf mit Ursachencode MQRC_OPTION_NOT_VALID_FOR_TYPE fehl und das Objekt wird nicht gelöscht.

Die Tabelle zeigt, welche Optionen zum Schließen von Objekten gültig sind und ob das Objekt beibehalten oder gelöscht wird.

Objekt- oder Warteschlangentyp	**MQCO_NONE**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Ein anderes Objekt als eine Warteschlange	Wird beibehalten	Ungültig	Ungültig
Vordefinierte Warteschlange	Wird beibehalten	Ungültig	Ungültig
Permanente dynamische Warteschlange	Wird beibehalten	Wird gelöscht, wenn sie leer ist und keine Aktualisierungen anstehen	Nachrichten werden gelöscht; Warteschlange wird gelöscht, wenn keine Aktualisierungen anstehen
Temporäre dynamische Warteschlange (Aufruf kommt vom Ersteller der Warteschlange)	Wird gelöscht	Wird gelöscht	Wird gelöscht

Die Tabelle zeigt, welche Optionen zum Schließen von Objekten gültig sind und ob das Objekt beibehalten oder gelöscht wird. *(Forts.)*			
Objekt- oder Warteschlangentyp	**MQCO_NONE**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Temporäre dynamische Warteschlange (Aufruf kommt nicht vom Ersteller der Warteschlange)	Wird beibehalten	Ungültig	Ungültig
Verteilerliste	Wird beibehalten	Ungültig	Ungültig
Verwaltetes Subskriptionsziel	Wird beibehalten	Ungültig	Ungültig
Verteilerliste (Subskription wurde entfernt)	Nachrichten werden gelöscht; Warteschlange wird gelöscht	Ungültig	Ungültig

Optionen zum Schließen von Subskriptionen: Diese Optionen steuern, ob permanente Subskriptionen entfernt werden, wenn die Kennung geschlossen wird, und ob Veröffentlichungen, die noch darauf warten, von der Anwendung gelesen zu werden, bereinigt werden. Diese Optionen sind nur für die Verwendung mit einer Objektkennung gültig, die mit dem Parameter *Hsub* eines MQSUB-Aufrufs zurückgegeben wird.

MQCO_KEEP_SUB
Die Kennung für die Subskription wird geschlossen, aber die eingerichtete Subskription wird beibehalten. Es werden weiter Veröffentlichungen an das in der Subskription angegebene Ziel gesendet. Diese Option ist nur gültig, wenn die Subskription mit der Option MQSO_DURABLE eingerichtet wurde.

MQCO_KEEP_SUB ist der Standardwert, wenn es sich um eine permanente Subskription handelt.

MQCO_REMOVE_SUB
Die Subskription wird entfernt und die Kennung für die Subskription geschlossen.

Der Parameter *Hobj* des Aufrufs MQSUB wird durch das Schließen des Parameters *Hsub* nicht ungültig gemacht und kann weiter für MQGET oder MQCB verwendet werden, um die übrigen Veröffentlichungen zu empfangen. Wenn der Parameter *Hobj* des Aufrufs MQSUB ebenfalls geschlossen wird und es sich um ein verwaltetes Ziel handelte, werden alle nicht abgerufenen Veröffentlichungen gelöscht.

MQCO_REMOVE_SUB ist der Standardwert, wenn es sich um eine nicht permanente Subskription handelt.

Die folgenden Tabellen enthalten eine Zusammenfassung der Optionen zum Schließen von Subskriptionen.

Um die Kennung für eine permanente Subskription zu schließen, aber die Subskription beizubehalten, können folgende Optionen zum Schließen von Subskriptionen verwendet werden:

Task	**Option zum Schließen einer Subskription**
Veröffentlichungen für eine mit MQOPEN geöffnete Kennung beibehalten	MQCO_KEEP_SUB
Veröffentlichungen für eine mit MQOPEN geöffnete Kennung entfernen	Aktion nicht zulässig
Veröffentlichungen für eine mit MQSO_MANAGED verwaltete Kennung beibehalten	MQCO_KEEP_SUB

Task	Option zum Schließen einer Subskription
Veröffentlichungen für eine mit MQSO_MANAGED verwaltete Kennung entfernen	Aktion nicht zulässig

Verwenden Sie die folgenden Optionen zum Schließen von Subskriptionen, um eine Subskription zu beenden, indem Sie entweder die Kennung einer permanenten Subskription schließen und die zugehörige Subskription aufheben oder indem Sie die Kennung einer nicht permanenten Subskription schließen:

Task	Option zum Schließen einer Subskription
Veröffentlichungen für eine mit MQOPEN geöffnete Kennung beibehalten	MQCO_REMOVE_SUB
Veröffentlichungen für eine mit MQOPEN geöffnete Kennung entfernen	Aktion nicht zulässig
Veröffentlichungen für eine mit MQSO_MANAGED verwaltete Kennung beibehalten	MQCO_REMOVE_SUB

Optionen für Vorauslesen: Die folgenden Optionen steuern, was mit nicht persistenten Nachrichten geschieht, die an den Client gesendet wurden, bevor eine Anwendung sie anforderte, und die noch nicht von der Anwendung verarbeitet wurden. Diese Nachrichten werden im Vorauslesepuffer des Clients gespeichert und warten darauf, von der Anwendung angefordert zu werden. Sie können entweder aus der Warteschlange gelöscht oder gelesen werden, bevor der MQCLOSE-Aufruf ausgeführt wird.

MQCO_IMMEDIATE
Das Objekt wird sofort geschlossen und alle Nachrichten, die an den Client gesendet wurden, bevor eine Anwendung sie anforderte, werden gelöscht und stehen der Anwendung nicht mehr zum Lesen zur Verfügung. Dies ist der Standardwert.

MQCO_QUIESCE
Es wird eine Anforderung zum Schließen des Objekts gestellt, aber wenn sich noch Nachrichten im Vorauslesepuffer des Clients befinden, die an den Client gesendet wurden, bevor eine Anwendung sie anforderte, gibt der MQCLOSE-Aufruf die Warnung MQRC_READ_AHEAD_MSGS zurück und die Objektkennung bleibt gültig.

Die Anwendung kann mit der Objektkennung weiter Nachrichten abrufen, bis keine mehr verfügbar sind, und das Objekt dann erneut schließen. Es werden jetzt nur noch Nachrichten an den Client gesendet, nachdem sie von einer Anwendung angefordert wurden. Die Vorauslesefunktion ist inaktiviert.

Es wird empfohlen, in Anwendungen die Option MQCO_QUIESCE zu verwenden, statt zu versuchen, einen Punkt zu erreichen, an dem sich keine Nachrichten mehr im Vorauslesepuffer des Clients befinden. Es kann nämlich passieren, dass zwischen dem letzten MQGET-Aufruf und dem folgenden MQCLOSE eine Nachricht eintrifft, die bei Verwendung der Option MQCO_IMMEDIATE gelöscht würde.

Wenn ein MQCLOSE mit MQCO_QUIESCE aus einer asynchronen Callback-Funktion ausgegeben wird, gilt beim Vorauslesen von Nachrichten dasselbe Verhalten. Wenn die Warnung MQRC_READ_AHEAD_MSGS zurückgegeben wird, wird die Callback-Funktion mindestens noch ein Mal aufgerufen. Sobald die letzte verbliebene Nachricht, die vorausgelesen wurde, an die Callback-Funktion übergeben wurde, wird das MQCBC-Feld ConsumerFlags auf MQCBCF_READA_BUFFER_EMPTY gesetzt.

Standardoption: Falls Sie keine der oben beschriebenen Optionen benötigen, können Sie folgende Option verwenden:

MQCO_NONE
Keine Option zum Schließen der Verarbeitung erforderlich.

Diese Option *muss* angegeben werden für:

- andere Objekte als Warteschlangen
- vordefinierte Warteschlangen
- temporäre dynamische Warteschlangen (aber nur, wenn *Hobj nicht* die Kennung ist, die von dem MQOPEN-Aufruf, der die Warteschlange erstellt hat, zurückgegeben wurde)
- Verteilerlisten

In allen oben genannten Fällen wird das Objekt beibehalten und nicht gelöscht.

Wenn diese Option für eine temporäre dynamische Warteschlange angegeben wird, gilt Folgendes:

- Die Warteschlange wird gelöscht, wenn sie von dem MQOPEN-Aufruf erstellt wurde, der *Hobj* zurückgegeben hat. Alle Nachrichten in der Warteschlange werden gelöscht.
- In allen anderen Fällen wird die Warteschlange (mit allen darin enthaltenen Nachrichten) beibehalten.

Bei Angabe dieser Option für eine permanente dynamische Warteschlange wird die Warteschlange beibehalten und nicht gelöscht.

Unter z/OS wird die Warteschlange physisch gelöscht, wenn es sich um eine dynamische Warteschlange handelt, die logisch gelöscht wurde, und dies die letzte Kennung für die Warteschlange ist. Weitere Informationen finden Sie im Abschnitt „Hinweise zur Verwendung” auf Seite 648.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Die aufgelisteten Ursachencodes kann der Warteschlangenmanager für den Parameter *Reason* zurückgeben.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Nachrichtengruppe nicht vollständig

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Logische Nachricht nicht vollständig

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC__DB2_NOT_AVAILABLE
(2342, X'926') Db2-Subsystem nicht verfügbar.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') MQOPEN- oder MQCLOSE-Aufruf: Option für Objekttyp ungültig

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_Q_NOT_EMPTY
(2055, X'807') Warteschlange enthält mindestens eine Nachricht oder nicht festgeschriebene PUT- oder GET-Anforderungen.

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx') Auf dem Client befinden sich vorausgelesene Nachrichten, die noch nicht von der Anwendung verarbeitet wurden

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SECURITY_ERROR
(2063, X'80F') Es ist ein Sicherheitsfehler aufgetreten.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Wenn eine Anwendung den MQDISC-Aufruf ausgibt oder entweder normal oder abnormal beendet wird, werden alle Objekte, die von der Anwendung geöffnet wurden und noch geöffnet sind, mit der Option MQCO_NONE automatisch geschlossen.
2. Wenn es sich bei dem zu schließenden Objekt um eine *Warteschlange* handelt, gilt Folgendes:
   - Wenn Operationen für die Warteschlange als Teil einer Arbeitseinheit ausgeführt werden, kann die Warteschlange geschlossen werden, bevor oder nachdem der Synchronisationspunkt eintritt, ohne dass sich dies auf das Ergebnis des Synchronisationspunkts auswirkt. Wenn die Warteschlange ausgelöst wird, kann die Ausführung eines Rollbacks vor dem Schließen der Warteschlange dazu führen, dass eine Auslösenachricht ausgegeben wird. Weitere Informationen zu Auslösenachrichten finden Sie im Abschnitt Eigenschaften von Auslösenachrichten.
   - Wenn die Warteschlange mit der Option MQOO_BROWSE geöffnet wurde, wird der Anzeigecursor gelöscht. Wird die Warteschlange anschließend erneut mit der Option MQOO_BROWSE geöffnet, wird ein neuer Anzeigecursor erstellt (siehe MQOO_BROWSE).
   - Wenn eine Nachricht zum Zeitpunkt des MQCLOSE-Aufrufs für diese Kennung gesperrt ist, wird die Sperre aufgehoben (siehe MQGMO_LOCK).
   - Wenn unter z/OS eine MQGET-Anforderung mit der Option MQGMO_SET_SIGNAL für die zu schließende Warteschlangenkennung ansteht, wird die Anforderung abgebrochen (siehe MQGMO_SET_SIGNAL). Signalanforderungen für dieselbe Warteschlange, die sich aber auf andere Kennungen (*Hobj*) beziehen, sind nicht betroffen (außer wenn eine dynamische Warteschlange gelöscht wird; dann werden auch sie abgebrochen).
3. Wenn es sich bei dem zu schließenden Objekt um eine *dynamische Warteschlange* (permanent oder temporär) handelt, gilt Folgendes:
   - Für eine dynamische Warteschlange können Sie die Optionen MQCO_DELETE und MQCO_DELETE_PURGE angeben. Dies ist unabhängig davon, welche Optionen im entsprechenden MQOPEN-Aufruf angegeben sind.
   - Beim Löschen einer dynamischen Warteschlange werden alle MQGET-Aufrufe mit der Option MQGMO_WAIT, die noch für die Warteschlange anstehen, abgebrochen und es wird der Ursachencode MQRC_Q_DELETED zurückgegeben. Siehe MQGMO_WAIT.

     Anwendungen können zwar nicht auf eine gelöschte Warteschlange zugreifen, aber die Warteschlange wird erst vom System entfernt und zugeordnete Ressourcen werden erst freigegeben, nachdem alle Kennungen, die auf die zu schließende Warteschlange verweisen, und alle Arbeitseinheiten, die die Warteschlange betreffen, entweder festgeschrieben oder zurückgesetzt wurden.

     Unter z/OS verhindert eine Warteschlange, die logisch gelöscht, aber noch nicht vom System entfernt wurde, die Erstellung einer neuen Warteschlange mit dem gleichen Namen wie dem der gelöschten Warteschlange. In diesem Fall schlägt der MQOPEN-Aufruf mit Ursachencode MQRC_NAME_IN_USE fehl. Eine solche Warteschlange kann auch weiterhin mit MQSC-Befehlen angezeigt werden, selbst wenn Anwendungen nicht darauf zugreifen können.
   - Wenn eine permanente dynamische Warteschlange gelöscht wird und es sich bei der *Hobj*-Kennung, die im MQCLOSE-Aufruf angegeben ist, *nicht* um die Kennung handelt, die von dem MQOPEN-Aufruf zurückgegeben wurde, der die Warteschlange erstellt hat, wird überprüft, ob die Benutzer-ID, die zur Auswertung des MQOPEN-Aufrufs verwendet wurde, zum Löschen der Warteschlange berechtigt ist. Wenn in dem MQOPEN-Aufruf die Option MQOO_ALTERNATE_USER_AUTHORITY angegeben wurde,

handelt es sich bei der überprüften Benutzer-ID um die alternative Benutzer-ID (*AlternateUserId*).

Diese Überprüfung findet in folgenden Fällen nicht statt:

– Die angegebene Kennung ist die Kennung, die von dem MQOPEN-Aufruf zurückgegeben wurde, der die Warteschlange erstellt hat.

– Die zu löschende Warteschlange ist eine temporäre dynamische Warteschlange.

- Wenn eine temporäre dynamische Warteschlange geschlossen wird und es sich bei der *Hobj*-Kennung, die im MQCLOSE-Aufruf angegeben ist, um die Kennung handelt, die von dem MQOPEN-Aufruf zurückgegeben wurde, der die Warteschlange erstellt hat, wird die Warteschlange gelöscht. Dies geschieht unabhängig davon, welche Optionen im MQCLOSE-Aufruf angegeben sind. Falls die Warteschlange Nachrichten enthält, werden sie gelöscht; es werden keine Berichtsnachrichten generiert.

  Wenn es nicht festgeschriebene Arbeitseinheiten gibt, die die Warteschlange betreffen, werden die Warteschlange und die darin enthaltenen Nachrichten trotzdem gelöscht, die Arbeitseinheiten schlagen jedoch nicht fehl. Allerdings werden, wie oben beschrieben, die den Arbeitseinheiten zugeordneten Ressourcen erst freigegeben, nachdem jede der Arbeitseinheiten entweder festgeschrieben oder zurückgesetzt wurde.

4. Wenn es sich bei dem zu schließenden Objekt um eine *Verteilerliste* handelt, gilt Folgendes:

   - Die einzige gültige Option zum Schließen einer Verteilerliste ist MQCO_NONE. Der Aufruf schlägt mit Ursachencode MQRC_OPTIONS_ERROR oder MQRC_OPTION_NOT_VALID_FOR_TYPE fehl, wenn eine andere Option angegeben wird.
   - Beim Schließen einer Verteilerliste werden für die Warteschlangen in der Liste keine einzelnen Beendigungscodes und Ursachencodes zurückgegeben. Zu Diagnosezwecken sind nur die Parameter *CompCode* und *Reason* des Aufrufs verfügbar.

     Wenn beim Schließen einer der Warteschlangen ein Fehler auftritt, setzt der Warteschlangenmanager die Verarbeitung fort und versucht, die übrigen Warteschlangen in der Verteilerliste zu schließen. Die Parameter *CompCode* und *Reason* des Aufrufs werden auf Rückkehrinformationen gesetzt, die den Fehler beschreiben. Es ist möglich, dass der Beendigungscode MQCC_FAILED lautet, obwohl die meisten Warteschlangen erfolgreich geschlossen wurden. Die Warteschlange, die den Fehler verursacht hat, wird nicht angegeben.

     Tritt bei mehreren Warteschlangen ein Fehler auf, ist nicht festgelegt, welcher Fehler in den Parametern *CompCode* und *Reason* zurückgemeldet wird.

5. Wenn unter IBM i die Verbindung der Anwendung implizit bei der ersten Ausgabe des MQOPEN-Aufrufs erfolgte, findet bei der letzten Ausgabe des MQCLOSE-Aufrufs ein impliziter MQDISC-Aufruf statt.

   Nur Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, können implizit verbunden werden; andere Anwendungen müssen den Aufruf MQCONN oder MQCONNX ausgeben, um explizit eine Verbindung mit dem Warteschlangenmanager herzustellen.

C-Aufruf

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
     CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Änderungen festschreiben

Der MQCMIT-Aufruf teilt dem Warteschlangenmanager mit, dass die Anwendung einen Synchronisationspunkt erreicht hat und dass alle seit dem letzten Synchronisationspunkt vorgenommenen Nachrichteneinreihungen und -abrufe permanent gespeichert werden sollen.

Nachrichten, die als Teil einer Arbeitseinheit eingereiht wurden, werden anderen Anwendungen verfügbar gemacht; Nachrichten, die als Teil einer Arbeitseinheit abgerufen wurden, werden gelöscht.

- Unter z/OS wird der Aufruf nur von Stapelverarbeitungsprogrammen verwendet (einschließlich IMS Stapel-DL/I-Programme).
- Unter IBM i wird dieser Aufruf nicht für Anwendungen unterstützt, die im Kompatibilitätsmodus ausgeführt werden.

Syntax

MQCMIT (*Hconn*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Die aufgelisteten Ursachencodes kann der Warteschlangenmanager für den Parameter *Reason* zurückgeben.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_BACKED_OUT
(2003, X'7D3') Arbeitseinheit zurückgesetzt

MQRC_OUTCOME_PENDING
(2124, X'84C') Ergebnis der Festschreibungsoperation ist anstehend

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT oder MQCMIT wurde unterbrochen und die Verbindungswiederholung kann kein definitives Ergebnis wiederherstellen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Aufruf in Umgebung nicht gültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OUTCOME_MIXED
(2123, X'84B') Ergebnis der Festschreibungs- oder Rücksetzungsoperation ist gemischt

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RECONNECT_FAILED
(2548, X'9F4') Nach der Wiederverbindung ist ein Fehler aufgetreten, der die Kennungen für eine wiederverbindbare Verbindung wiedereingesetzt hat

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Verwenden Sie diesen Aufruf nur, wenn der Warteschlangenmanager selbst die Arbeitseinheit koordiniert. Dieser kann Folgendes einschließen:
   - Eine lokale Arbeitseinheit, bei der die Änderungen nur WebSphere MQ-Ressourcen betreffen.
   - Eine globale Arbeitseinheit, bei der die Änderungen neben den WebSphere MQ-Ressourcen auch Ressourcen anderer Ressourcenmanager betreffen können.

   Nähere Details über lokale und globale Arbeitseinheiten finden Sie im Abschnitt „MQBEGIN - Arbeitseinheit starten” auf Seite 623.
2. In Umgebungen, in denen die Arbeitseinheit nicht durch den Warteschlangenmanager koordiniert wird, muss anstelle von MQCMIT der entsprechende Commit-Aufruf zum Festschreiben verwendet werden. Die Umgebung unterstützt möglicherweise auch eine implizite Festschreibung, die durch ein normales Beenden der Anwendung verursacht wird.
   - Verwenden Sie auf z/OS folgende Aufrufe:
     – Stapelverarbeitungsprogramme (einschließlich IMS Stapel-DL/I-Programme) können den MQCMIT-Aufruf verwenden, wenn sich die Arbeitseinheit nur auf WebSphere MQ-Ressourcen

auswirkt. Wenn sich die Arbeitseinheit allerdings sowohl auf WebSphere MQ-Ressourcen als auch auf Ressourcen anderer Ressourcenmanager (z. B. DB2) auswirkt, verwenden Sie den SRRCMIT-Aufruf, der vom z/OS Recoverable Resource Service (RRS) bereitgestellt wird. Der SRRCMIT-Aufruf schreibt Änderungen anderer Ressourcenmanager fest, die für RRS-Koordination aktiviert wurden.

- CICS -Anwendungen müssen den Befehl `EXEC CICS SYNCPOINT` verwenden, um die Arbeitseinheit explizit festzuschreiben. Andernfalls führt das Beenden der Transaktion zu einer impliziten Festschreibung der Arbeitseinheit. Der MQCMIT-Aufruf kann nicht bei CICS-Anwendungen verwendet werden.
- Bei IMS-Anwendungen (außer Stapel-DL/I-Programme) müssen Sie IMS-Aufrufe wie `GU` und `CHKP` verwenden, um die Arbeitseinheit festzuschreiben. Der MQCMIT-Aufruf kann nicht bei IMS-Anwendungen (außer Stapel-DL/I-Programme) verwendet werden.

- Unter IBM i verwenden Sie diesen Aufruf für lokale Arbeitseinheiten, die vom Warteschlangenmanager koordiniert werden. Dies bedeutet, dass auf Jobebene keine COMMIT-Definition existieren darf und somit der Befehl `STRCMTCTL` mit dem Parameter `CMTSCOPE(*JOB)` für den Job nicht ausgegeben worden sein darf.

3. Wird eine Anwendung mit nicht festgeschriebenen Änderungen in einer Arbeitseinheit beendet, ist die Verwerfung dieser Änderungen davon abhängig, ob die Anwendung auf normale oder nicht normale Weise beendet wird. Weitere Informationen finden Sie im Abschnitt Hinweise zur Verwendung von MQDISC.
4. Wenn eine Anwendung Nachrichten in Gruppen oder Segmenten logischer Nachrichten einreiht oder abruft, speichert der Warteschlangenmanager die Informationen über die Nachrichtengruppe und logische Nachricht für die letzten erfolgreichen MQPUT- und MQGET-Aufrufe. Diese Informationen sind mit der Warteschlangenkennung verknüpft und umfassen Folgendes:

   - Die Werte der Felder *GroupId*, *MsgSeqNumber*, *Offset* und *MsgFlags* in MQMD
   - Ist die Nachricht Teil einer Arbeitseinheit
   - Bei einem MQPUT-Aufruf: ob die Nachricht persistent oder nicht persistent ist.

   Wenn eine Arbeitseinheit festgeschrieben ist, behält der Warteschlangenmanager die Gruppen- und Segmentinformationen und die Anwendung kann weiterhin Nachrichten in die aktuelle Nachrichtengruppe oder logische Nachricht einreihen oder daraus abrufen.

   Durch das Beibehalten der früheren Werte der Gruppen- und Segmentinformationen beim Festschreiben einer Arbeitseinheit kann die Anwendung eine große Nachrichtengruppe oder eine aus vielen Segmenten bestehende große logische Nachricht über mehrere Arbeitseinheiten verteilen. Das Verwenden mehrerer Arbeitseinheiten ist vorteilhaft, wenn der lokale Warteschlangenmanager nur beschränkten Warteschlangenspeicher hat. Die Anwendung muss allerdings ausreichend Informationen beibehalten, um das Einreihen und Abrufen von Nachrichten an der korrekten Position neu zu starten, wenn ein Systemausfall auftritt. Weitere Informationen zum Neustart an der korrekten Position nach einem Systemausfall finden Sie unter MQPMO_LOGICAL_ORDER und MQGMO_LOGICAL_ORDER.

   Die weiteren Hinweise gelten nur, wenn die Koordination der Arbeitseinheiten durch den Warteschlangenmanager erfolgt:

5. Eine Arbeitseinheit hat denselben Geltungsbereich wie eine Verbindungskennung. Alle WebSphere MQ-Aufrufe, die eine bestimmte Arbeitseinheit betreffen, müssen mit derselben Verbindungskennung ausgeführt werden. Mit einer anderen Verbindungskennung ausgegebene Aufrufe (zum Beispiel Aufrufe durch eine andere Anwendung) betreffen eine andere Arbeitseinheit. Informationen zum Geltungsbereich von Verbindungskennungen finden Sie in der Beschreibung des Parameters *Hconn* im Abschnitt MQCONN.
6. Dieser Aufruf wirkt sich nur auf Nachrichten aus, die als Teil der aktuellen Arbeitseinheit eingereiht oder abgerufen wurden.
7. Eine Anwendung mit langer Laufzeit, die MQGET-, MQPUT- oder MQPUT1-Aufrufe in einer Arbeitseinheit ausgibt, aber nie einen Festschreibungs- oder Rücksetzungsaufruf ausgibt, kann Warteschlangen mit Nachrichten anfüllen, die für andere Anwendungen nicht verfügbar sind. Um sich davor zu schützen, muss der Administrator das Warteschlangenmanagerattribut *MaxUncommittedMsgs* auf einen Wert festlegen, der niedrig genug ist, um ein Auffüllen der Warteschlangen durch solche Anwendun-

gen zu verhindern, aber hoch genug, damit die erwarteten Messaging-Anwendungen ordnungsgemäß arbeiten können.

8. Auf UNIX- und Windows-Systemen ist es möglich, dass die Arbeitseinheit erfolgreich festgeschrieben wurde, wenn der Parameter *Reason* MQRC_CONNECTION_BROKEN (mit einem *CompCode* von MQCC_FAILED) oder MQRC_UNEXPECTED_ERROR angibt.

C-Aufruf

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Aufruf von High Level Assembler

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Aufruf in Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCONN - Warteschlangenmanager verbinden

Der MQCONN-Aufruf verbindet ein Anwendungsprogramm mit einem Warteschlangenmanager.

Er stellt eine Warteschlangenmanager-Verbindungskennung bereit, die von der Anwendung bei nachfolgenden Message-Queuing-Aufrufen verwendet wird.

- Unter z/OS müssen CICS-Anwendungen diesen Aufruf nicht ausgeben. Diese Anwendungen werden automatisch mit dem Warteschlangenmanager verbunden, mit dem das CICS-System verbunden ist. Die MQCONN- und MQDISC-Aufrufe werden allerdings von CICS-Anwendungen akzeptiert.
- Unter IBM i müssen Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, diesen Aufruf nicht ausgeben. Diese Anwendungen werden automatisch mit dem Warteschlangenmanager verbunden, wenn sie den ersten MQOPEN-Aufruf ausgeben. Die MQCONN- und MQDISC-Aufrufe werden allerdings von IBM i-Anwendungen akzeptiert.

  Andere Anwendungen (d. h. Anwendungen, die nicht im Kompatibilitätsmodus ausgeführt werden) müssen den MQCONN- oder MQCONNX-Aufruf ausgeben, um eine Verbindung mit dem Warteschlangenmanager herzustellen, und den MQDISC-Aufruf, um die Verbindung zum Warteschlangenmanager zu trennen. Dies ist die empfohlene Art der Programmierung.

Eine Clientverbindung kann nicht auf einer reinen Serverinstallation und eine lokale Verbindung nicht auf einer reinen Clientinstallation hergestellt werden.

Syntax

MQCONN (*QMgrName*, *Hconn*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Dies ist der Name des Warteschlangenmanagers, mit dem die Anwendung eine Verbindung herstellen will. Der Name kann folgende Zeichen enthalten:

- Großbuchstaben (A bis Z)
- Kleinbuchstaben des Alphabets(a bis z)
- Numerische Ziffern (0 bis 9)
- Punkt (.), Schrägstrich (/), Unterstrich (_), Prozent (%)

Der Name darf keine führenden oder eingebetteten Leerzeichen enthalten, wohl aber abschließende Leerzeichen. Als Endekennzeichen für die signifikanten Daten im Namen kann ein Nullzeichen verwendet werden; die Null und alle nachfolgenden Zeichen werden als Leerzeichen behandelt. Die folgenden Beschränkungen gelten für die angegebenen Umgebungen:

- Bei Systemen, die EBCDIC Katakana verwenden, können keine Kleinbuchstaben verwendet werden.
- Unter z/OS können Namen, die mit einem Unterstrich beginnen oder enden, nicht von den Operationen und Steuerkonsolen verarbeitet werden. Aus diesem Grund sind solche Namen zu vermeiden.
- Unter IBM i setzen Sie Namen, die Kleinbuchstaben, Schrägstriche oder Prozentzeichen enthalten, in Anführungszeichen, wenn sie in Befehlen angegeben werden. Geben Sie diese Anführungszeichen nicht im Parameter *QMgrName* an.

Wenn der Name nur aus Leerzeichen besteht, wird der Name des *Standard*-Warteschlangenmanagers verwendet.

Der für *QMgrName* angegebene Name muss der Name eines Warteschlangenmanagers sein, mit dem eine *Verbindung hergestellt werden kann*.

Unter z/OS werden die Warteschlangenmanager, mit denen eine Verbindung herstellt werden kann, durch die Umgebung festgelegt:

- In CICS können Sie nur den Warteschlangenmanager verwenden, mit dem das CICS-System verbunden ist. Der Parameter *QMgrName* muss angegeben werden, aber sein Wert wird ignoriert. Es wird empfohlen, Leerzeichen zu verwenden.
- In IMS kann nur mit Warteschlangenmanagern eine Verbindung hergestellt werden, die in der Subsystem-Definitionstabelle (CSQQDEFV) *und* in der SSM-Tabelle in IMS aufgelistet sind (siehe Hinweis 6).
- Unter z/OS Stapel und TSO kann nur mit Warteschlangenmanagern eine Verbindung hergestellt werden, die sich im selben System wie die Anwendung befinden (siehe Hinweis 6).

Gruppen mit gemeinsamer Warteschlange:Auf Systemen mit mehreren Warteschlangenmanagern, die als Gruppe mit gemeinsamer Warteschlange konfiguriert sind, kann für *QMgrName* der Name der Gruppe mit gemeinsamer Warteschlange statt des Namens eines Warteschlangenmanagers angegeben werden. Dies ermöglicht es der Anwendung, mit einem *beliebigen* Warteschlangenmanager eine Verbindung herzustellen, der in der Gruppe mit gemeinsamer Warteschlange verfügbar ist und sich auf demselben z/OS-Image wie die Anwendung befindet. Das System kann auch so konfiguriert werden, dass bei Verwendung eines leeren *QMgrName* eine Verbindung zur Gruppe mit gemeinsamer Warteschlange anstatt zum Standardwarteschlangenmanager hergestellt wird.

Wenn *QMgrName* den Namen einer Gruppe mit gemeinsamer Warteschlange angibt, es aber auch einen Warteschlangenmanager desselben Namens auf dem System gibt, wird die Verbindung zu diesem hergestellt. Nur wenn diese Verbindung fehlschlägt, wird versucht, eine Verbindung zu einem der Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange herzustellen.

Bei erfolgreichem Verbindungsaufbau können Sie die vom MQCONN- oder MQCONNX-Aufruf zurückgegebene Kennung verwenden, um auf *alle* Ressourcen (sowohl gemeinsam genutzte als auch nicht gemeinsam genutzte) zuzugreifen, die zum Warteschlangenmanager gehören, zu dem die Verbindung hergestellt wurde. Der Zugriff auf diese Ressourcen unterliegt den typischen Berechtigungsprüfungen.

Wenn die Anwendung zwei MQCONN- oder MQCONNX-Aufrufe ausgibt, um gleichzeitig bestehende Verbindungen aufzubauen, und mindestens einer der Aufrufe den Namen der Gruppe mit gemeinsamer Warteschlange angibt, gibt der zweite Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_ALREADY_CONNECTED zurück, wenn er eine Verbindung mit demselben Warteschlangenmanager wie der erste Aufruf herstellt.

Gruppen mit gemeinsamer Warteschlange werden nur unter z/OS unterstützt. Die Verbindung zu einer Gruppe mit gemeinsamer Warteschlange wird nur in den Stapel-, RRS-Stapel- und TSO-Umgebungen unterstützt.

WebSphere MQ MQI-Clientanwendungen: Bei WebSphere MQ MQI-Clientanwendungen wird der Verbindungsaufbau für jede Clientverbindungskanaldefinition mit dem angegebenen Warteschlangenmanagernamen versucht, bis eine Verbindung erfolgreich ist. Der Warteschlangenmanager muss allerdings denselben Namen wie der angegebene Name haben. Wenn ein Name mit Leerzeichen angegeben wird, wird jeder Clientverbindungskanal mit einem Warteschlangenmanagernamen mit Leerzeichen ausprobiert, bis eine Verbindung erfolgreich ist. In diesem Fall wird der tatsächliche Name des Warteschlangenmanagers nicht geprüft.

WebSphere MQ-Clientanwendungen werden unter z/OS nicht unterstützt, aber z/OS kann als WebSphere MQ-Server agieren, zu dem WebSphere MQ-Clientanwendungen eine Verbindung herstellen können.

WebSphere MQ MQI-Client-Warteschlangenmanagergruppen: Wenn der angegebene Name mit einem Stern (*) beginnt, kann der Warteschlangenmanager, zu dem die Verbindung hergestellt wird, einen anderen Namen als den von der Anwendung angegebenen haben. Der angegebene Name (ohne Stern) definiert eine *Gruppe* von Warteschlangenmanagern, die für eine Verbindung infrage kommen. Zur Auswahl probiert die Implementierung alle Warteschlangenmanager nacheinander, bis einer gefunden wird, mit dem die Verbindung hergestellt werden kann. Die Reihenfolge, in der die Herstellung

einer Verbindung versucht wird, wird von der Wertigkeit des Clientkanals und den Verbindungsaffinitätswerten der potenziellen Kanäle beeinflusst. Wenn keiner der Warteschlangenmanager in der Gruppe verfügbar ist, schlägt der Aufruf fehl. Jeder Warteschlangenmanager wird nur einmal probiert. Wenn ein einzelner Stern für den Namen angegeben wird, wird eine implementierungsspezifische Warteschlangenmanagergruppe verwendet.

WS-Manager-Gruppen werden nur für Anwendungen unterstützt, die in einer MQ-Clientumgebung ausgeführt werden. Der Aufruf schlägt fehl, wenn eine Nicht-Clientanwendung einen mit einem Stern beginnenden WS-Manager-Namen angibt. Eine Gruppe wird definiert, indem mehrere Clientverbindungskanaldefinitionen mit demselben WS-Manager-Namen (angegebener Name ohne Stern) bereitgestellt werden, um mit jedem Warteschlangenmanager in der Gruppe zu kommunizieren. Die Standardgruppe wird definiert, indem eine oder mehrere Clientverbindungskanaldefinitionen bereitgestellt werden, jede mit einem leeren Warteschlangenmanagernamen (die Angabe eines Namens mit Leerzeichen hat daher dieselbe Auswirkung wie die Angabe eines einzelnen Sterns für den Namen einer Clientanwendung).

Wenn die Verbindung zu einem Warteschlangenmanager einer Gruppe hergestellt ist, kann eine Anwendung Leerzeichen auf übliche Art in den Warteschlangenmanagernamensfeldern in Nachricht- und Objektdeskriptoren angeben, um den Warteschlangenmanager zu benennen, mit dem die Anwendung verbunden ist (der *lokale Warteschlangenmanager*). Wenn die Anwendung diesen Namen kennen muss, verwenden Sie den MQINQ-Aufruf, um das WS-Manager-Attribut *QMgrName* abzufragen.

Dem Verbindungsnamen einen Stern voranzustellen impliziert, dass die Anwendung nicht von einer Verbindung zu einem bestimmten Warteschlangenmanager in der Gruppe abhängt. Geeignete Anwendungen sind:

- Anwendungen, die Nachrichten einreihen, aber keine abrufen.
- Anwendungen, die Anforderungsnachrichten einreihen und dann die Antwortnachrichten aus einer *temporären dynamischen* Warteschlange abrufen.

Nicht geeignet sind Anwendungen, die Nachrichten aus einer bestimmten Warteschlange bei einem bestimmten Warteschlangenmanager abrufen müssen. Diese Anwendungen dürfen dem Namen keinen Stern voranstellen.

Wenn Sie einen Stern angeben, ist die maximale Länge für den Rest des Namens 47 Zeichen.

Warteschlangenmanagergruppen werden unter z/OS nicht unterstützt.

Die Länge dieses Parameters wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

Hconn

Typ: MQHCONN - Ausgabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Geben Sie sie bei allen nachfolgenden Message-Queuing-Aufrufen an, die von der Anwendung ausgegeben werden. Die Kennung wird ungültig, wenn der #MQDISC-Aufruf ausgegeben oder die Verarbeitungseinheit, die den Geltungsbereich der Kennung definiert, beendet wird.

WebSphere MQ stellt jetzt die MQM-Bibliothek mit Clientpaketen und Serverpaketen bereit. Das bedeutet, bei einem MQI-Aufruf, der in der MQM-Bibliothek gefunden wird, wird der Verbindungstyp darauf überprüft, ob es sich um eine Client- oder eine Serververbindung handelt, und dann der korrekte zugrundeliegende Aufruf ausgegeben. Dadurch kann ein Exit, dem *Hconn* übergeben wird, jetzt mit der MQM-Bibliothek verknüpft, aber auf einer Clientinstallation verwendet werden.

Geltungsbereich der Kennung:Der Geltungsbereich der zurückgegebenen Kennung hängt vom Aufruf ab, der für die Verbindung zum Warteschlangenmanager verwendet wird (MQCONN oder MQCONNX). Wenn der verwendete Aufruf MQCONNX ist, hängt der Geltungsbereich der Kennung auch von der Option MQCNO_HANDLE_SHARE_* ab, die im Feld *Options* der MQCNO-Struktur angegeben ist.

- Wenn der Aufruf MQCONN ist oder die Option MQCNO_HANDLE_SHARE_NONE angegeben ist, ist die zurückgegebene Kennung eine *nicht gemeinsam genutzte* Kennung.

  Der Geltungsbereich einer nicht gemeinsam genutzten Kennung ist die kleinste Parallelverarbeitungseinheit, die von der Plattform unterstützt wird, auf der die Anwendung ausgeführt wird (Details

finden Sie unter Tabelle 564 auf Seite 658). Die Kennung ist außerhalb der Parallelverarbeitungseinheit, von der der Aufruf abgesetzt wurde, nicht gültig.

- Wenn Sie die Option MQCNO_HANDLE_SHARE_BLOCK oder MQCNO_HANDLE_SHARE_NO_BLOCK angeben, ist die zurückgegebene Kennung eine *gemeinsam genutzte* Kennung.

  Der Geltungsbereich einer gemeinsam genutzten Kennung ist der Prozess, der Eigner des Threads ist, von dem der Aufruf ausgegeben wurde. Die Kennung kann von jedem Thread dieses Prozesses verwendet werden. Nicht alle Plattformen unterstützen Threads.
- Wenn der MQCONN- oder MQCONNX-Aufruf mit einem Beendigungscode gleich MQCC_FAILED fehlschlägt, ist der Wert "Hconn" nicht definiert.

Tabelle 564. Geltungsbereich nicht gemeinsam genutzter Kennungen auf verschiedenen Plattformen

Plattform	**Geltungsbereich nicht gemeinsam genutzter Kennungen**
z/OS	• CICS: die CICS-Task • IMS: die Task bis zum nächsten Synchronisationspunkt (ausgenommen Subtasks der Task) • z/OS-Stapel und TSO: die Task (ausgenommen Subtasks der Task)
IBM i	Job
UNIX-Systeme	Thread
16-Bit-Windows-Anwendungen	Prozess
32-Bit-Windows-Anwendungen	Thread

Bei CICS-Anwendungen unter z/OSund Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, lautet der zurückgegebene Wert:

MQHC_DEF_HCONN
Standardverbindungskennung

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') Anwendung bereits verbunden

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') Laden des Exits für Clusterauslastung nicht möglich

MQRC_SSL_ALREADY_INITIALIZED
(2391, X'957') SSL bereits initialisiert

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851') Adapterverbindungsmodul kann nicht geladen werden

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853') Adaptersubsystem-Definitionsmodul ungültig

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854') Adaptersubsystem-Definitionsmodul kann nicht geladen werden

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ADAPTER_STORAGE_SHORTAGE
(2127, X'84F') Speicherknappheit bei Adapter

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837') Anderer Warteschlangenmanager bereits verbunden

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_INIT_ERROR
(2375, X'947') API-Exit-Initialisierung fehlgeschlagen

MQRC_API_EXIT_TERM_ERROR
(2376, X'948') API-Exit-Abschluss fehlgeschlagen

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONN_ID_IN_USE
(2160, X'870') Verbindungs-ID bereits im Gebrauch

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_ERROR
(2273, X'8E1') Fehler bei der Verarbeitung des MQCONN-Anrufs.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Tritt bei einem MQCONN- oder MQCONNX-Aufruf auf, wenn der Warteschlangenmanager eine Verbindung des angeforderten Verbindungstyps auf der aktuellen Installation nicht bereitstellen kann. Eine Clientverbindung kann nicht auf einer Serverinstallation hergestellt werden. Eine lokale Verbindung kann nicht auf einer Clientinstallation hergestellt werden.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Konfigurationsfehler bei Verschlüsselungshardware

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873') Wiederherstellungskoordinator vorhanden

MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Aufruf in Umgebung nicht gültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Von einem Client wurde ein MQCONN-Aufruf zur Herstellung der Verbindung mit einem Warteschlangenmanager ausgegeben, aber der Versuch, die Kommunikation mit dem fernen System herzustellen, ist fehlgeschlagen.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Keine Übereinstimmung zwischen Warteschlangenmanagerinstallation und ausgewählter Bibliothek

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Schlüsselrepository ungültig

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Maximale Anzahl Verbindungen erreicht

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_OPEN_FAILED
(2137, X'859') Objekt nicht erfolgreich geöffnet

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SECURITY_ERROR
(2063, X'80F') Es ist ein Sicherheitsfehler aufgetreten.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X'959') SSL-Initialisierungsfehler

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Der Warteschlangenmanager, zu dem eine Verbindung mit einem MQCONN-Aufrufs hergestellt wird, wird *lokaler Warteschlangenmanager* genannt.
2. Warteschlangen im Eigentum des lokalen Warteschlangenmanagers erscheinen gegenüber den Anwendungen als lokale Warteschlangen. Es ist möglich, Nachrichten in diese Warteschlangen einzureihen und sie von ihnen abzurufen.

   Gemeinsam genutzte Warteschlangen im Eigentum der Gruppe mit gemeinsamer Warteschlange, zu welcher der lokale Warteschlangenmanager gehört, erscheinen gegenüber der Anwendung als lokale Warteschlangen. Es ist möglich, Nachrichten in diese Warteschlangen einzureihen und sie von ihnen abzurufen.

   Warteschlangen im Eigentum ferner Warteschlangenmanager erscheinen als ferne Warteschlangen. Es ist möglich, Nachrichten in diese Warteschlangen einzureihen, aber nicht, Nachrichten aus ihnen abzurufen.

3. Wenn der Warteschlangenmanager ausfällt, während eine Anwendung ausgeführt wird, muss die Anwendung erneut den MQCONN-Aufruf ausgeben, um eine neue Verbindungskennung für die Verwendung bei nachfolgenden WebSphere MQ-Aufrufen zu erhalten. Die Anwendung kann den MQCONN-Aufruf in regelmäßigen Abständen ausgeben, bis er erfolgreich ausgeführt wird.

   Wenn eine Anwendung nicht erkennt, ob sie mit dem Warteschlangenmanager verbunden ist, kann sie problemlos einen MQCONN-Aufruf ausgeben, um eine Verbindungskennung zu erhalten. Wenn die Anwendung bereits verbunden ist, wird dieselbe Kennung wie beim vorherigen MQCONN-Aufruf zurückgegeben, aber mit Beendigungscode MQCC_WARNING und Ursachencode MQRC_ALREADY_CONNECTED.
4. Wenn die Anwendung die Verwendung von WebSphere MQ-Aufrufen abgeschlossen hat, muss sie die Verbindung zum Warteschlangenmanager mit dem MQDISC-Aufruf beenden.
5. Wenn der MQCONN-Aufruf mit einem Beendigungscode gleich MQCC_FAILED fehlschlägt, ist der Wert "Hconn" nicht definiert.
6. Unter z/OS:

   - Stapel-, TSO- und IMS-Anwendungen müssen den MQCONN-Aufruf ausgeben, um die anderen WebSphere MQ-Aufrufe zu verwenden. Diese Anwendungen können jeweils zu mehr als einem Warteschlangenmanager eine Verbindung herstellen.

     Wenn der Warteschlangenmanager ausfällt, muss die Anwendung den Aufruf nach dem Warteschlangenmanagerneustart erneut ausgeben, um eine neue Verbindungskennung zu erhalten.

     Auch wenn IMS-Anwendungen den MQCONN-Aufruf wiederholt ausgeben können, selbst wenn sie bereits verbunden sind, wird dies bei Online-Nachrichenverarbeitungsprogrammen (MPPs) nicht empfohlen.
   - CICS-Anwendungen müssen den MQCONN-Aufruf nicht ausgeben, um die anderen WebSphere MQ-Aufrufe zu verwenden, können dies aber tun; sowohl der MQCONN-Aufruf als auch der MQDISC-Aufruf werden akzeptiert. Es ist allerdings nicht möglich, zu jeweils mehr als einem Warteschlangenmanager eine Verbindung herzustellen.

     Wenn der Warteschlangenmanager ausfällt, werden diese Anwendungen nach einem Warteschlangenmanagerneustart automatisch wieder verbunden und müssen daher den MQCONN-Aufruf nicht ausgeben.
7. Unter z/OS definieren Sie die verfügbaren Warteschlangenmanager folgendermaßen:

   - Bei Stapelanwendungen können Systemprogrammierer das Makro CSQBDEF verwenden, um ein Modul (CSQBDEFV) zu erstellen, das den Standard-Warteschlangenmanagernamen oder den Namen der Gruppe mit gemeinsamer Warteschlange definiert.
   - Bei IMS-Anwendungen können Systemprogrammierer das Makro CSQQDEFX verwenden, um ein Modul (CSQQDEFV) zu erstellen, das die Namen von verfügbaren Warteschlangenmanagern definiert und den Standardwarteschlangenmanager angibt.

     Zusätzlich muss jeder Warteschlangenmanager für die IMS-Steuerregion und für jede abhängige Region definiert werden, die auf diesen Warteschlangenmanager zugreifen. Um dies zu erreichen, müssen Sie eine Subsystem-Teildatei in der IMS.PROCLIB-Bibliothek erstellen und die Subsystem-Teildatei in den entsprechenden IMS-Bereichen angeben. Wenn eine Anwendung versucht, zu einem Warteschlangenmanager eine Verbindung herzustellen, der nicht in der Subsystem-Teildatei für die IMS-Region definiert ist, wird die Anwendung abnormal beendet.
8. Unter IBM i können Anwendungen, die für Vorgängerreleases des Warteschlangenmanagers geschrieben wurden, ohne Neukompilierung ausgeführt werden. Dies wird*Kompatibilitätsmodus* genannt. Dieser Betriebsmodus stellt eine kompatible Laufzeitumgebung für Anwendungen bereit. Der Kompatibilitätsmodus umfasst:

   - Das Serviceprogramm AMQZSTUB in der Bibliothek QMQM.

     AMQZSTUB bietet die gleiche allgemein zugängliche Schnittstelle wie frühere Releases und besitzt die gleiche Signatur. Verwenden Sie dieses Serviceprogramm, um auf MQI mit gebundenen Prozeduraufrufen zuzugreifen.

- Das Programm QMQM in der Bibliothek QMQM.

  QMQM bietet eine Möglichkeit für MQI-Zugriffe mittels dynamischer Programmaufrufe.
- Die Programme MQCLOSE, MQCONN, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQPUT1, und MQSET in der Bibliothek QMQM.

  Diese Programme ermöglichen es ebenfalls, auf MQI mit dynamischen Programmaufrufen zuzugreifen, aber mit einer Parameterliste, die den Standard-Beschreibungen der WebSphere MQ-Aufrufe entspricht.

Diese drei Schnittstellen beziehen nicht die Funktionalität ein, die in WebSphere MQ Version 5.1 eingeführt wurde. Die Aufrufe MQBACK, MQCMIT und MQCONNX werden beispielsweise nicht unterstützt. Für diese Schnittstellen werden ausschließlich Anwendungen mit nur einem Thread unterstützt.

Unterstützung für die neuen WebSphere MQ-Aufrufe in Einzelthread-Anwendungen und für alle WebSphere MQ-Aufrufe in Multithread-Anwendungen wird durch die Serviceprogramme LIBMQM und LIBMQM_R bereitgestellt.

9. Unter IBM i werden Programme, die abnormal enden, nicht automatisch vom Warteschlangenmanager getrennt. Schreiben Sie Anwendungen, die es dem MQCONN- oder MQCONNX-Aufruf ermöglichen, den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_ALREADY_CONNECTED zurückzugeben. Verwenden Sie die in dieser Situation zurückgegebene Verbindungskennung als normal.

C-Aufruf

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN   Hconn;     /* Connection handle */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN     DS  F     Connection handle
COMPCODE  DS  F     Completion code
REASON    DS  F     Reason code qualifying COMPCODE
```

Aufruf in Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 – Verbindung mit Warteschlangenmanager herstellen (erweitert)

Der MQCONNX-Aufruf verbindet ein Anwendungsprogramm mit einem Warteschlangenmanager. Er stellt eine WS-Manager-Verbindungskennung bereit, die von der Anwendung bei nachfolgenden WebSphere MQ-Aufrufen verwendet wird.

Der MQCONNX-Aufruf entspricht dem MQCONN-Aufruf, außer dass MQCONNX die Angabe von Optionen zur Steuerung der Ausführung des Aufrufs ermöglicht.

- Dieser Aufruf wird von allen WebSphere MQ-Systemen sowie von WebSphere MQ-Clients, die mit diesen Systemen verbunden sind, unterstützt.
- Unter IBM i wird dieser Aufruf nicht für Anwendungen unterstützt, die im Kompatibilitätsmodus ausgeführt werden.

Eine Clientverbindung kann nicht auf einer reinen Serverinstallation und eine lokale Verbindung nicht auf einer reinen Clientinstallation hergestellt werden.

Syntax

MQCONNX (*QMgrName*, *ConnectOpts*, *Hconn*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Details siehe Beschreibung des Parameters *QMgrName* im Abschnitt „MQCONN - Warteschlangenmanager verbinden” auf Seite 655.

ConnectOpts
Typ: MQCNO - Ein-/Ausgabe

Weitere Informationen finden Sie im Abschnitt „MQCNO - Verbindungsoptionen” auf Seite 300.

Hconn
Typ: MQHCONN - Ausgabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Geben Sie sie bei allen nachfolgenden Message-Queuing-Aufrufen an, die von der Anwendung ausgegeben werden. Die Kennung wird ungültig, wenn der #MQDISC-Aufruf ausgegeben oder die Verarbeitungseinheit, die den Geltungsbereich der Kennung definiert, beendet wird.

WebSphere MQ stellt jetzt die MQM-Bibliothek mit Clientpaketen und Serverpaketen bereit. Das bedeutet, bei einem MQI-Aufruf, der in der MQM-Bibliothek gefunden wird, wird der Verbindungstyp darauf überprüft, ob es sich um eine Client- oder eine Serververbindung handelt, und dann der korrekte zugrundeliegende Aufruf ausgegeben. Dadurch kann ein Exit, dem *Hconn* übergeben wird, jetzt mit der MQM-Bibliothek verknüpft, aber auf einer Clientinstallation verwendet werden.

Geltungsbereich der Kennung:Der Geltungsbereich der zurückgegebenen Kennung hängt vom Aufruf ab, der für die Verbindung zum Warteschlangenmanager verwendet wird (MQCONN oder MQCONNX). Wenn der verwendete Aufruf MQCONNX ist, hängt der Geltungsbereich der Kennung auch von der Option MQCNO_HANDLE_SHARE_* ab, die im Feld *Options* der MQCNO-Struktur angegeben ist.

- Wenn der Aufruf MQCONN ist oder die Option MQCNO_HANDLE_SHARE_NONE angegeben ist, ist die zurückgegebene Kennung eine *nicht gemeinsam genutzte* Kennung.

  Der Geltungsbereich einer nicht gemeinsam genutzten Kennung ist die kleinste Parallelverarbeitungseinheit, die von der Plattform unterstützt wird, auf der die Anwendung ausgeführt wird (Details finden Sie unter Tabelle 565 auf Seite 664). Die Kennung ist außerhalb der Parallelverarbeitungseinheit, von der der Aufruf abgesetzt wurde, nicht gültig.
- Wenn Sie die Option MQCNO_HANDLE_SHARE_BLOCK oder MQCNO_HANDLE_SHARE_NO_BLOCK angeben, ist die zurückgegebene Kennung eine *gemeinsam genutzte* Kennung.

  Der Geltungsbereich einer gemeinsam genutzten Kennung ist der Prozess, der Eigner des Threads ist, von dem der Aufruf ausgegeben wurde. Die Kennung kann von jedem Thread dieses Prozesses verwendet werden. Nicht alle Plattformen unterstützen Threads.
- Wenn der MQCONN- oder MQCONNX-Aufruf mit einem Beendigungscode gleich MQCC_FAILED fehlschlägt, ist der Wert "Hconn" nicht definiert.

Tabelle 565. Geltungsbereich nicht gemeinsam genutzter Kennungen auf verschiedenen Plattformen

Plattform	**Geltungsbereich nicht gemeinsam genutzter Kennungen**
z/OS	• CICS: die CICS-Task • IMS: die Task bis zum nächsten Synchronisationspunkt (ausgenommen Subtasks der Task) • z/OS-Stapel und TSO: die Task (ausgenommen Subtasks der Task)
IBM i	Job
UNIX-Systeme	Thread
16-Bit-Windows-Anwendungen	Prozess
32-Bit-Windows-Anwendungen	Thread

Bei CICS-Anwendungen unter z/OSund Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, lautet der zurückgegebene Wert:

MQHC_DEF_HCONN
Standardverbindungskennung

CompCode
Typ: MQLONG - Ausgabe

Details siehe Beschreibung des Parameters *CompCode* im Abschnitt „MQCONN - Warteschlangenmanager verbinden” auf Seite 655.

Reason

Typ: MQLONG - Ausgabe

Die folgenden Codes können von den Aufrufen MQCONN und MQCONNX zurückgegeben werden. Im Folgenden werden die zusätzlichen Codes aufgelistet, die vom Aufruf MQCONNX zurückgegeben werden können.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') Anwendung bereits verbunden

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') Laden des Exits für Clusterauslastung nicht möglich

MQRC_SSL_ALREADY_INITIALIZED
(2391, X'957') SSL bereits initialisiert

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851') Adapterverbindungsmodul kann nicht geladen werden

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853') Adaptersubsystem-Definitionsmodul ungültig

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854') Adaptersubsystem-Definitionsmodul kann nicht geladen werden

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ADAPTER_STORAGE_SHORTAGE
(2127, X'84F') Speicherknappheit bei Adapter

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837') Anderer Warteschlangenmanager bereits verbunden

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_INIT_ERROR
(2375, X'947') API-Exit-Initialisierung fehlgeschlagen

MQRC_API_EXIT_TERM_ERROR
(2376, X'948') API-Exit-Abschluss fehlgeschlagen

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONN_ID_IN_USE
(2160, X'870') Verbindungs-ID bereits im Gebrauch

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_ERROR
(2273, X'8E1') Fehler bei der Verarbeitung des MQCONN-Anrufs.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Tritt bei einem MQCONN- oder MQCONNX-Aufruf auf, wenn der Warteschlangenmanager eine Verbindung des angeforderten Verbindungstyps auf der aktuellen Installation nicht bereitstellen kann. Eine Clientverbindung kann nicht auf einer Serverinstallation hergestellt werden. Eine lokale Verbindung kann nicht auf einer Clientinstallation hergestellt werden.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Konfigurationsfehler bei Verschlüsselungshardware

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873') Wiederherstellungskoordinator vorhanden

MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Aufruf in Umgebung nicht gültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Von einem Client wurde ein MQCONN-Aufruf zur Herstellung der Verbindung mit einem Warteschlangenmanager ausgegeben, aber der Versuch, die Kommunikation mit dem fernen System herzustellen, ist fehlgeschlagen.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Keine Übereinstimmung zwischen Warteschlangenmanagerinstallation und ausgewählter Bibliothek

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Schlüsselrepository ungültig

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Maximale Anzahl Verbindungen erreicht

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_OPEN_FAILED
(2137, X'859') Objekt nicht erfolgreich geöffnet

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SECURITY_ERROR
(2063, X'80F') Es ist ein Sicherheitsfehler aufgetreten.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X'959') SSL-Initialisierungsfehler

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Folgende zusätzlichen Ursachencodes können vom MQCONNX-Aufruf zurückgegeben werden:

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_AIR_ERROR
(2385, X'951') Authentifizierungsdatensatz ungültig.

MQRC_AUTH_INFO_CONN_NAME_ERROR
(2387, X'953') Name der Authentifizierungsdatenverbindung ungültig.

MQRC_AUTH_INFO_REC_COUNT_ERROR
(2383, X'94F') Authentifizierungsdatensatzzähler ungültig.

MQRC_AUTH_INFO_REC_ERROR
(2384, X'950') Authentifizierungsdatensatzfelder ungültig.

MQRC_AUTH_INFO_TYPE_ERROR
(2386, X'952') Authentifizierungsdatentyp ungültig.

MQRC_CD_ERROR
(2277, X'8E5') Kanaldefinition ungültig.

MQRC_CLIENT_CONN_ERROR
(2278, X'8E6') Clientverbindungsfelder ungültig

MQRC_CNO_ERROR
(2139, X'85B') Verbindungsoptionsstruktur ungültig

MQRC_CONN_TAG_IN_USE
(2271, X'8DF') Verbindungskennung belegt.

MQRC_CONN_TAG_NOT_USABLE
(2350, X'92E') Verbindungskennung nicht verwendbar.

MQRC_LDAP_PASSWORD_ERROR
(2390, X'956') LDAP-Kennwort ungültig.

MQRC_LDAP_USER_NAME_ERROR
(2388, X'954') LDAP-Benutzernamensfelder ungültig.

MQRC_LDAP_USER_NAME_LENGTH_ERR
(2389, X'955') LDAP-Benutzernamenslänge ungültig.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_SCO_ERROR
(2380, X'94C') SSL-Konfigurationsoptionsstruktur ungültig.

MQRC_SSL_CONFIG_ERROR
(2392, X'958') SSL-Konfigurationsfehler.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

Für die Programmiersprache Visual Basic gilt Folgendes:

- Der Parameter *ConnectOpts* wird als Typ MQCNO deklariert. Wenn die Anwendung als MQI-Client von WebSphere MQ ausgeführt wird und Sie die Parameter des Clientverbindungskanals angeben möchten, deklarieren Sie den Parameter *ConnectOpts* als Typ Any, sodass in der Anwendung eine MQCNOCD-Struktur anstelle einer MQCNO-Struktur im Aufruf angegeben werden kann. Dies bedeutet jedoch, dass der Parameter *ConnectOpts* nicht überprüft werden kann, um sicherzustellen, dass er den richtigen Datentyp hat.

C-Aufruf

```
MQCONNX (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQCONNX' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCONNX (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQCONNX,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQCONNX QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Nachrichtenkennung erstellen

Der Aufruf MQCRTMH gibt eine Nachrichtenkennung zurück.

Eine Anwendung kann den Aufruf MQCRTMH in nachfolgenden Message-Queuing-Aufrufen verwenden:

- Über den MQSETMP-Aufruf können Sie eine Eigenschaft der Nachrichtenkennung festlegen.
- Über den MQINQMP-Aufruf können Sie den Wert einer Eigenschaft der Nachrichtenkennung abfragen.
- Über den MQDLTMP-Aufruf können Sie eine Eigenschaft der Nachrichtenkennung löschen.

Die Nachrichtenkennung kann im Aufruf MQPUT und MQPUT1 verwendet werden, um die Eigenschaften der Nachrichtenkennung mit denen der Nachricht zu verknüpfen, die eingereiht wird. Ähnlich kann durch Angabe einer Nachrichtenkennung im Aufruf MQGET auf die Eigenschaften der Nachricht, die abgerufen wird, mit der Nachrichtenkennung zugegriffen werden, wenn der Aufruf MQGET abgeschlossen wurde.

Über MQDLTMH können Sie eine Nachrichtenkennung löschen.

Syntax

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben. Ist die Verbindung zum Warteschlangenmanager nicht mehr gültig und wird an der Nachrichtenkennung kein WebSphere MQ-Aufruf ausgeführt, wird MQDLTMH implizit aufgerufen, um die Nachricht zu löschen.

Sie können auch den folgenden Wert angeben:

MQHC_UNASSOCIATED_HCONN
Die Verbindungskennung stellt keine Verbindung zu einem bestimmten Warteschlangenmanager dar.

Wird dieser Wert verwendet, muss die Nachrichtenkennung durch einen expliziten Aufruf von MQDLTMH gelöscht werden, um den ihr zugeordneten Speicherplatz freizugeben; von WebSphere MQ wird die Nachrichtenkennung in keinem Fall implizit gelöscht.

Es muss mindestens eine gültige Verbindung zu einem in dem Thread, der die Nachrichtenkennung erstellt, erstellten Warteschlangenmanager bestehen. Andernfalls schlägt der Aufruf mit MQRC_HCONN_ERROR fehl.

In einer Umgebung mit mehreren Installationen auf einem einzigen System wird der Wert MQHC_UNASSOCIATED_HCONN auf die Verwendung mit der ersten im Prozess geladenen Installation beschränkt. Der Ursachencode MQRC_HMSG_NOT_AVAILABLE wird zurückgegeben, wenn die Nachrichtenkennung für eine andere Installation angegeben wird.

Für CICS-Anwendungen unter z/OS und Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, kann der Aufruf MQCONN übergangen und folgender Wert für *Hconn* angegeben werden:

MQHC_DEF_CONN
Standardverbindungskennung

CrtMsgHOpts
Typ: MQCMHO - Eingabe

Die Optionen zur Steuerung der Aktion von MQCRTMH. Details hierzu finden Sie im Abschnitt MQCMHO.

Hmsg
Typ: MQHMSG - Ausgabe

Bei der Ausgabe wird eine Nachrichtenkennung zurückgegeben, mit der deren Eigenschaften festgelegt, abgefragt und gelöscht werden können. Zunächst hat die Nachrichtenkennung keine Eigenschaften.

Außerdem ist der Nachrichtenkennung ein Nachrichtendeskriptor zugeordnet. Dieser enthält zunächst die Standardwerte. Die Werte der zugehörigen Nachrichtendeskriptorfelder können mit den Aufrufen MQSETMP und MQINQMP festgelegt und abgefragt werden. Der Aufruf MQDLTMP setzt ein Feld des Nachrichtendeskriptors zurück auf den Standardwert.

Wenn der Parameter *Hconn* als der Wert MQHC_UNASSOCIATED_HCONN angegeben ist, kann die zurückgegebene Nachrichtenkennung in MQGET-, MQPUT- oder MQPUT1-Aufrufen mit einer beliebigen Verbindung innerhalb der Verarbeitungseinheit verwendet werden. Die Kennung kann jedoch jeweils nur von einem WebSphere MQ-Aufruf verwendet werden. Wenn die Kennung verwendet wird, wenn ein zweiter WebSphere MQ-Aufruf versucht, dieselbe Nachrichtenkennung zu verwenden, schlägt der zweite WebSphere MQ-Aufruf mit dem Ursachencode MQRC_MSG_HANDLE_IN_USE fehl.

Wenn der Parameter *Hconn* nicht den Wert MQHC_UNASSOCIATED_HCONN hat, kann die zurückgegebene Nachrichtenkennung nur in der angegebenen Verbindung verwendet werden.

Derselbe Parameterwert für *Hconn* muss in den nachfolgenden MQI-Aufrufen verwendet werden, in denen diese Nachrichtenkennung verwendet wird:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

Die zurückgegebene Nachrichtenkennung verliert ihre Gültigkeit, wenn der MQDLTMH-Aufruf für die Nachrichtenkennung ausgegeben oder die Verarbeitungseinheit, die den Bereich der Kennung definiert, beendet wird. MQDLTMH wird implizit aufgerufen, wenn eine bestimmte Verbindung bei der Erstellung der Nachrichtenkennung bereitgestellt wird und die Verbindung zum Warteschlangenmanager ihre Gültigkeit verliert, beispielsweise wenn MQDBC aufgerufen wird.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CMHO_ERROR
(2461, X'099D') Struktur Optionen Nachrichtenkennung erstellen nicht gültig.

MQRC_CONNECTION_BROKEN
(2273, X'7D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'07E1') Keine weiteren Kennungen verfügbar.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennungsverweis ungültig.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;       /* Connection handle */
MQCMHO   CrtMsgHOpts; /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;        /* Message handle */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

COBOL

```
 CALL 'MQCRTMH' USING HCONN, CRTMSGOPTS, HMSG, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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);
```

Deklarieren Sie die Parameter wie folgt:

```
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)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Callbacks steuern

Der MQCTL-Aufruf führt Steuerungsaktionen für Callbacks und die für eine Verbindung geöffneten Objektkennungen aus.

Syntax

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Ursache*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS -Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen werden und Sie können den folgenden Sonderwert für *Hconn*angeben:

MQHC_DEF_HCONN
Standardverbindungskennung

Operation
Typ: MQLONG - Eingabe

Die Operation, die für den Callback verarbeitet wird, die für die angegebene Objektkennung definiert ist. Sie müssen eine einzige der folgenden Optionen angeben:

MQOP_START

Startet die Verarbeitung von Nachrichten für alle definierten Nachrichtenkonsumentenfunktionen für die angegebene Verbindungskennung.

Callbacks werden in einem vom System gestarteten Thread ausgeführt, der sich von allen Anwendungs-Threads unterscheidet.

Diese Operation ermöglicht die Steuerung der bereitgestellten Verbindungskennung für das System. Die einzigen MQI-Aufrufe, die von einem anderen Thread als dem Konsumententhread ausgegeben werden können, sind:

- MQCTL mit Operation MQOP_STOP
- MQCTL mit Operation MQOP_SUSPEND
- MQDISC - Führt MQCTL mit der Operation MQOP_STOP aus, bevor es die Verbindung zu HConn trennt.

MQRC_HCONN_ASYNC_ACTIVE wird zurückgemeldet, wenn ein WebSphere MQ API-Aufruf ausgegeben wird, während die Verbindungskennung gestartet wird und wenn der Aufruf nicht von einer Nachrichtenkonsumentenfunktion stammt.

Stoppt ein Nachrichtenkonsument die Verbindung während des MQCBCT_START_CALL, meldet der MQCTL-Aufruf den Fehlerursachencode MQRC_CONNECTION_STOPPED zurück.

Die Ausgabe kann über eine Konsumentenfunktion erfolgen. Für dieselbe Verbindung wie die der Callback-Routine dient diese nur dazu, eine zuvor ausgegebene MQOP_STOP-Operation zu stornieren.

Diese Option wird in den folgenden Umgebungen nicht unterstützt: CICS on z/OS oder wenn die Anwendung mit einer WebSphere MQ-Bibliothek ohne Thread gebunden ist.

MQOP_START_WAIT
Startet die Verarbeitung von Nachrichten für alle definierten Nachrichtenkonsumentenfunktionen für die angegebene Verbindungskennung.

Nachrichtenkonsumenten werden im selben Thread ausgeführt und die Kontrolle wird erst an den Aufrufer von MQCTL zurückgegeben:

- Freigabe durch Verwendung der Operation MQCTL MQOP_STOP oder MQOP_SUSPEND, oder
- Registrierung aller Konsumentenroutinen zurückgenommen oder ausgesetzt wurde.

Wenn die Registrierung aller Konsumenten zurückgenommen oder ausgesetzt wurde, wird eine implizite MQOP_STOP-Operation ausgegeben.

Diese Option kann, weder für die aktuelle noch für eine andere Verbindungskennung, nicht innerhalb einer Callback-Routine verwendet werden. Bei einem Aufrufversuch wird MQRC_ENVIRONMENT_ERROR zurückgemeldet.

Wenn zu irgendeinem Zeitpunkt während einer MQOP_START_WAIT-Operation keine registrierten, nicht ausgesetzten Konsumenten vorhanden sind, schlägt der Aufruf mit dem Ursachencode MQRC_NO_CALLBACKS_ACTIVE fehl.

Wenn die Verbindung während einer MQOP_START_WAIT-Operation ausgesetzt wird, meldet der MQCTL-Aufruf den Ursachencode MQRC_CONNECTION_SUSPENDED für die Warnung zurück; die Verbindung bleibt "gestartet".

Die Anwendung kann wahlweise MQOP_STOP oder MQOP_RESUME ausgeben. In diesem Fall wird die MQOP_RESUME-Operation blockiert.

Diese Option wird in einem Client mit Einzelthread nicht unterstützt.

MQOP_STOP
Stoppt die Verarbeitung von Nachrichten und wartet, bis alle Konsumenten ihre Operationen durchgeführt haben, bevor diese Option ausgeführt wird. Diese Operation gibt die Verbindungskennung frei.

Wird diese Option innerhalb einer Callback-Routine ausgeführt, wirkt sie sich erst nach Beendigung der Routine aus. Es werden keine Nachrichtenkonsumentenroutinen mehr aufgerufen, nachdem die Konsumentenroutinen für bereits gelesene Nachrichten abgeschlossen sind und Stop-Aufrufe (falls angefordert) für Callback-Routinen getätigt wurden.

Erfolgt die Ausgabe außerhalb einer Callback-Routine, wird die Kontrolle dem Aufrufer erst zurückgegeben, wenn die Konsumentenroutinen für bereits gelesene Nachrichten und an Callbacks gesendete Aufrufe zum Beenden (sofern angefordert) ausgeführt wurden. Die Callbacks selbst bleiben dagegen registriert.

Diese Funktion wirkt sich nicht auf Vorauslesenachrichten aus. Sie müssen sicherstellen, dass Konsumenten MQCLOSE(MQCO_QUIESCE) aus der Callback-Funktion heraus ausführen, um festzustellen, ob weitere auszuliefernde Nachrichten vorhanden sind.

MQOP_SUSPEND
Hält die Verarbeitung von Nachrichten an. Diese Operation gibt die Verbindungskennung frei.

Dies hat keine Auswirkungen auf das Vorauslesen von Nachrichten für die Anwendung. Wenn Sie die Verarbeitung von Nachrichten für längere Zeit stoppen möchten, überlegen Sie, die Warteschlange zu schließen und erneut zu öffnen, wenn die Verarbeitung fortgesetzt wird.

Wird diese Option innerhalb einer Callback-Routine ausgeführt, wirkt sie sich erst nach Beendigung der Routine aus. Es werden keine weiteren Nachrichtenkonsumentenroutinen aufgerufen, nachdem die aktuelle Routine beendet wurde.

Erfolgt der Aufruf außerhalb eines Callback, wird die Kontrolle dem Aufrufer erst zurückgegeben, wenn die aktuelle Konsumentenroutine ausgeführt wurde und keine weitere aufgerufen wird.

MQOP_RESUME
Setzt die Verarbeitung von Nachrichten fort.

Diese Option wird normalerweise im Thread der Hauptanwendung ausgeführt. Sie kann aber auch in einer Callback-Routine eingesetzt werden, um eine frühere Aussetzungsanforderung aufzuheben, die in derselben Routine ausgegeben wurde.

Wird MQOP_RESUME verwendet, um ein MQOP_START_WAIT fortzusetzen, wird die Operation blockiert.

ControlOpts
Typ: MQCTLO - Eingabe

Optionen zur Steuerung der Aktion von MQCTL

Details zur Struktur finden Sie im Abschnitt „MQCTLO – Struktur der Optionen für die Callback-Steuerung” auf Seite 319.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') Servicemodule für Datenkonvertierung können nicht geladen werden.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Callback-Routine kann nicht aufgerufen werden

MQRC_CALLBACK_NOT_ REGISTERED
(2448, X'990') Aufheben der Registrierung, Aussetzen oder Fortsetzen nicht möglich, da kein registrierter Callback vorhanden ist

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Es wurden entweder sowohl CallbackFunction als auch CallbackName in einem MQOP_REGISTER-Aufruf angegeben.

Oder es wurde entweder CallbackFunction oder CallbackName angegeben, die aber nicht mit der aktuell registrierten Callback-Funktion übereinstimmen.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Feld CallBackType falsch.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CBD_ERROR
(2444, X'98C') Falscher Optionsblock.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Falsches Feld für MQCBD-Optionen.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Fehler bei Korrelations-ID.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') Die angeforderte Funktion ist in der aktuellen Umgebung nicht verfügbar.

MQRC_GET_INHIBITED
(2016, X'7E0') wird für die Warteschlange unterdrückt.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Konflikt mit globaler Arbeitseinheit.

MQRC_GMO_ERROR
(2186, X'88A') Optionsstruktur zum Nachrichtenabruf ungültig.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Kennung für globale Arbeitseinheit belegt.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') Spezifikation zum Durchsuchen inkonsistent.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Nachricht unter Cursor nicht gültig für Abruf

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') Globale Arbeitseinheit im Konflikt mit lokaler Arbeitseinheit.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Abgleichoptionen ungültig

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B5') Falsches Feld für MaxMsgLength.

MQRC_MD_ERROR
(2026, X'7EA') Nachrichtendeskriptor ungültig

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') Der Funktionseingangspunkt konnte im Modul nicht gefunden werden.

MQRC_MODULE_INVALID
(2496, X'9C0') Modul gefunden, allerdings hat es den falschen Typ (32 Bit/64 Bit) oder ist keine gültige DLL-Datei.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Modul im Suchpfad nicht gefunden oder keine Berechtigung zum Laden.

MQRC_MSG_ID_ERROR
(2206, X'89E') Fehler bei Nachrichten-ID

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Nachrichtenfolgenummer ungültig

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Verwendung des Nachrichtentokens ungültig.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Warteschlange nicht für Anzeige geöffnet

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Warteschlange nicht zur Eingabe geöffnet.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OPERATION_ERROR
(2488, X'9B8') Falscher Operationscode für API-Aufruf.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Warteschlange hat falschen Indextyp

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Signal für diese Kennung ausstehend.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') Unterstützung für Synchronisationspunkt nicht verfügbar.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Eintragung in globale Arbeitseinheit fehlgeschlagen.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') Mischung von Aufrufen für Arbeitseinheiten wird nicht unterstützt.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') Arbeitseinheit nicht zur Verwendung durch den Warteschlangenmanager verfügbar.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') Warteintervall in MQGMO ungültig

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Falsche MQGMO-Version bereitgestellt.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Falsche MQMD-Version bereitgestellt.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Callback-Routinen müssen die Antworten aller Services überprüfen, die sie aufrufen, und wenn die Routine einen Zustand erkennt, der nicht behoben werden kann, muss sie den Befehl MQCB MQOP_DEREGISTER ausgeben, um wiederholte Aufrufe der Callback-Routine zu verhindern.
2. Unter z/OS, bei der Operation MQOP_START:
   - Programme, die asynchrone Callback-Routinen verwenden, müssen für die Verwendung von z/OS UNIX System Services (USS) berechtigt sein.
   - Language Environment (LE) Programme, die asynchrone Callback-Routinen verwenden, müssen die LE-Laufzeitoption POSIX(ON) verwenden.
   - Andere als LE-Programme, die asynchrone Callback-Routinen verwenden, dürfen die USS-Schnittstelle pthread_create (aufrufbarer Service BPX1PTC) nicht verwenden.
3. MQCTL wird im IMS -Adapter nicht unterstützt.

Anmerkung: In CICS wird MQOP_START nicht unterstützt. Verwenden Sie stattdessen den Funktionsaufruf MQOP_START_WAIT.

C-Aufruf

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Verbindung mit Warteschlangenmanager beenden

Der MQDISC-Aufruf unterbricht die Verbindung zwischen dem Warteschlangenmanager und dem Anwendungsprogramm und ist die Umkehrfunktion des MQCONN- oder MQCONNX-Aufrufs.

- Unter z/OS muss bei allen Anwendungen, die asynchrone Nachrichtenverarbeitung, Ereignisverarbeitung oder Callback verwenden, der Hauptsteuerungsthread einen MQDISC-Aufruf vor dem Beenden ausgeben. Weitere Einzelheiten finden Sie unter Asynchrone Verarbeitung von WebSphere MQ-Nachrichten.
- Unter z/OS müssen CICS-Anwendungen diesen Aufruf nicht ausgeben, um die Verbindung zum Warteschlangenmanager zu trennen, aber möglicherweise, um die Verwendung eines Verbindungstags zu beenden.

- Unter IBM i müssen Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, diesen Aufruf nicht ausgeben. Weitere Informationen finden Sie im Abschnitt „MQCONN - Warteschlangenmanager verbinden” auf Seite 655.

Syntax

MQDISC (*Hconn*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Ein-/Ausgabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS-Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, können Sie den MQCONN-Aufruf übergehen und folgende Werte für *Hconn* angeben:

MQHC_DEF_HCONN
Standardverbindungskennung

Bei erfolgreicher Beendigung des Aufrufs setzt der Warteschlangenmanager *Hconn* auf einen Wert, der keine gültige Kennung für die Umgebung darstellt. Dieser Wert lautet:

MQHC_UNUSABLE_HCONN
Unbrauchbare Verbindungskennung

Unter z/OS wird *Hconn* auf einen Wert gesetzt, der nicht definiert ist.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_BACKED_OUT
(2003, X'7D3') Arbeitseinheit zurückgesetzt

MQRC_CONN_TAG_NOT_RELEASED
(2344, X'928') Verbindungstag nicht freigegeben

MQRC_OUTCOME_PENDING
(2124, X'84C') Ergebnis der Festschreibungsoperation ist anstehend

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_DISC_LOAD_ERROR
(2138, X'85A') Adapter-Verbindungsabbaumodul kann nicht geladen werden

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_INIT_ERROR
(2375, X'947') API-Exit-Initialisierung fehlgeschlagen

MQRC_API_EXIT_TERM_ERROR
(2376, X'948') API-Exit-Abschluss fehlgeschlagen

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_OUTCOME_MIXED
(2123, X'84B') Ergebnis der Festschreibungs- oder Rücksetzungsoperation ist gemischt

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Wenn ein MQDISC-Aufruf ausgegeben wird und die Verbindung noch geöffnete Objekte enthält, schließt der Warteschlangenmanager diese Objekte, bei denen die Schließoption auf MQCO_NONE festgelegt ist.
2. Wenn die Anwendung mit nicht festgeschriebenen Änderungen in einer Arbeitseinheit beendet wird, ist die Disposition dieser Änderungen davon abhängig, wie die Anwendung beendet wird.

   a. Wenn die Anwendung vor dem Beenden den MQDISC-Aufruf ausgibt:

   - Bei einer vom Warteschlangenmanager koordinierten Arbeitseinheit gibt der Warteschlangenmanager den MQCMIT-Aufruf für die Anwendung aus. Die Arbeitseinheit wird festgeschrieben, falls möglich, oder zurückgesetzt.

- Bei einer extern koordinierten Arbeitseinheit wird der Status der Arbeitseinheit nicht geändert, der Warteschlangenmanager gibt allerdings üblicherweise an, dass die Arbeitseinheit festgeschrieben werden muss, wenn er vom Arbeitseinheitenkoordinator gefragt wird.

  Unter z/OS, CICS, IMS (außer Stapel-DL/1-Programme) und RRS verhalten sich Anwendungen auf diese Weise.

b. Wenn die Anwendung normal beendet wird, ohne jedoch den MQDISC-Aufruf auszugeben, hängt die durchgeführte Aktion von der Umgebung ab:

- Unter z/OS werden, außer bei MQ Java- oder MQ JMS-Anwendungen, die unter Hinweis 2a beschriebenen Aktionen durchgeführt.
- In allen anderen Fällen werden die unter Hinweis 2c beschriebenen Aktionen durchgeführt.

In Anbetracht der Unterschiede in den Umgebungen sollten Sie sicherstellen, dass Anwendungen, die Sie portieren wollen, die Arbeitseinheit festschreiben oder zurücksetzen, bevor sie beendet werden.

c. Wenn die Anwendung *abnormal* beendet wird, ohne den MQDISC-Aufruf auszugeben, wird die Arbeitseinheit zurückgesetzt.

3. Unter z/OS gelten die folgenden Punkte:

- CICS-Anwendungen müssen den MQDISC-Aufruf nicht ausgeben, um die Verbindung zum Warteschlangenmanager zu trennen, weil das CICS-System selbst die Verbindung zum Warteschlangenmanager herstellt und der MQDISC-Aufruf keine Auswirkung auf diese Verbindung hat.
- CICS-, IMS- (außer Stapel-DL/1-Programme) und RRS-Anwendungen verwenden Arbeitseinheiten, die von einem externen Arbeitseinheitenkoordinator koordiniert werden. Daher wirkt sich der MQDISC-Aufruf nicht auf den Status der Arbeitseinheit (falls vorhanden) aus, die existiert, wenn der Aufruf ausgegeben wird.

  Der MQDISC-Aufruf *zeigt* allerdings das Nutzungsende des Verbindungstags *ConnTag* an, das der Verbindung durch einen früheren von der Anwendung ausgegebenen MQCONNX-Aufruf zugeordnet wurde. Falls es eine aktive Arbeitseinheit gibt, die den Verbindungstag referenziert, wenn der MQDISC-Aufruf ausgegeben wird, wird der Aufruf mit Beendigungscode MQCC_WARNING und Ursachencode MQRC_CONN_TAG_NOT_RELEASED beendet. Das Verbindungstag ist für die Wiederverwendung erst verfügbar, wenn der externe Arbeitseinheitenkoordinator die Arbeitseinheit aufgelöst hat.

4. Unter IBM i müssen Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, diesen Aufruf nicht ausgeben. Weitere Informationen finden Sie im Abschnitt zum MQCONN-Aufruf.

Anmerkung: In CICS wird MQOP_START nicht unterstützt. Verwenden Sie stattdessen den Funktionsaufruf MQOP_START_WAIT.

C-Aufruf

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

System/390 -Assembleraufruf

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Aufruf in Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQDLTMH - Nachrichtenkennung löschen

Der MQDLTMH-Aufruf löscht eine Nachrichtenkennung und ist die Umkehrfunktion des MQCRTMH-Aufrufs.

Syntax

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager.

Der Wert muss mit der Verbindungskennung übereinstimmen, die für die Erstellung der im Parameter *Hmsg* angegebenen Verbindungskennung verwendet wurde.

Wenn die Nachrichtenkennung mit MQHC_UNASSOCIATED_HCONN erstellt wurde, muss eine gültige Verbindung in dem Thread hergestellt werden, der die Nachrichtenkennung löscht. Andernfalls schlägt der Aufruf mit MQRC_CONNECTION_BROKEN fehl.

Hmsg
Typ: MQHMSG - Ein-/Ausgabe

Dies ist die zu löschende Nachrichtenkennung. Der Wert wurde durch einen früheren MQCRTMH-Aufruf zurückgemeldet.

Bei erfolgreicher Ausführung des Aufrufs wird die Kennung auf einen ungültigen Wert für die Umgebung gesetzt. Dieser Wert lautet:

MQHM_UNUSABLE_HMSG
Unbrauchbare Nachrichtenkennung.

Die Nachrichtenkennung kann nicht gelöscht werden, wenn ein anderer WebSphere MQ-Aufruf in Bearbeitung ist, an den dieselbe Nachrichtenkennung übergeben wurde.

DltMsgHOpts
Typ: MQDMHO - Eingabe

Weitere Informationen finden Sie im Abschnitt „MQDMHO – Optionen zum Löschen von Nachrichtenkennungen” auf Seite 338.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_DMHO_ERROR
(2462, X'099E') Struktur Optionen Nachrichtenkennung löschen ungültig.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennungsverweis ungültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C-Aufruf

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGOPTS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
** Connection handle
01  HCONN  PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01  DLTMSGHOPTS.
   COPY CMQDLMHOV.

** Completion code
01 COMPCODE  PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Aufruf in PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl DltMsgHOpts like MQDMHO;    /* Options that control the action of MQDLTMH */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

Aufruf von High Level Assembler

```
CALL MQDLTMH,(HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Löschen von Nachrichteneigenschaften

Der MQDLTMP-Aufruf löscht eine Eigenschaft aus einer Nachrichtenkennung und ist die Umkehrfunktion des MQSETMP-Aufrufs.

Syntax

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Name*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert muss mit der Verbindungskennung übereinstimmen, die für die Erstellung der im Parameter *Hmsg* angegebenen Verbindungskennung verwendet wurde.

Wenn die Nachrichtenkennung mit MQHC_UNASSOCIATED_HCONN erstellt wurde, muss eine gültige Verbindung in dem Thread erstellt werden, der die Nachrichtenkennung löscht. Andernfalls schlägt der Aufruf mit MQRC_CONNECTION_BROKEN fehl.

Hmsg
Typ: MQHMSG - Eingabe

Dies ist die Nachrichtenkennung mit der zu löschenden Eigenschaft. Der Wert wurde durch einen früheren MQCRTMH-Aufruf zurückgemeldet.

DltPropOpts
Typ: MQDMPO - Eingabe

Details hierzu finden Sie unter dem MQDMPO-Datentyp.

Name
Typ: MQCHARV - Eingabe

Der Name der zu löschenden Eigenschaft. Weitere Informationen zu Eigenschaftsnamen finden Sie im Abschnitt Eigenschaftsnamen.

Platzhalter sind im Eigenschaftsnamen nicht zulässig.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Eigenschaft nicht verfügbar.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975') Ein MQRFH2-Ordner, der Eigenschaften enthält, konnte nicht analysiert werden.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'086D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_DMPO_ERROR
(2481, X'09B1') Struktur der Optionen zum Löschen von Nachrichteneigenschaften nicht gültig.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennung nicht gültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Ungültiger Eigenschaftsname.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') ID des codierten Zeichensatzes des Eigenschaftsnamens nicht gültig.

MQRC_UNEXPECTED_ERROR
(2195, X'0893') Unerwarteter Fehler aufgetreten.

Detaillierte Informationen zu diesen Codes finden Sie in folgenden Abschnitten:

- Ursachencodes für WebSphere MQ for z/OS
- API-Ursachencodes für andere WebSphere MQ-Plattformen

.

C-Aufruf

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```

CALL MQDLTMP,(HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```

HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
DLTPROPOPTS  CMQDMPOA   ,     Options that control the action of MQDLTMP
NAME         CMQCHRVA   ,     Property name
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQGET - Nachricht abrufen

Der MQGET-Aufruf ruft eine Nachricht aus einer lokalen Warteschlange ab, die mit dem MQOPEN-Aufruf geöffnet wurde.

Syntax

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS- Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und der folgende Wert für *Hconn*angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Hobj
Typ: MQHOBJ - Eingabe

Diese Kennung steht für die Warteschlange, aus der eine Nachricht abgerufen werden soll. Der Wert von *Hobj* wurde von einem vorherigen MQOPEN-Aufruf zurückgegeben. Die Warteschlange muss mit einer der folgenden Optionen geöffnet worden sein (Details enthält der Abschnitt „MQOPEN – Objekt öffnen" auf Seite 728):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Diese Struktur beschreibt die Attribute der erforderlichen Nachricht und die der abgerufenen Nachricht. Weitere Informationen finden Sie im Artikel „MQMD - Nachrichtendeskriptor" auf Seite 399.

Wenn *BufferLength* kleiner als die Nachrichtenlänge ist, wird *MsgDesc* vom Warteschlangenmanager aufgefüllt, wenn MQGMO_ACCEPT_TRUNCATED_MSG beim Parameter *GetMsgOpts* angegeben ist (siehe MQGMO - Optionsfeld).

Wenn die Anwendung einen MQMD der Version 1 bereitstellt, hat die zurückgegebene Nachricht eine den Anwendungsnachrichtendaten vorangestellte MQMDE, aber *nur*, wenn ein oder mehrere Felder in der MQMDE einen Nicht-Standardwert haben. Wenn alle Felder in der MQMDE Standardwerte haben, wird die MQMDE weggelassen. Ein Formatname von MQFMT_MD_EXTENSION im Feld *Format* im MQMD gibt an, dass eine MQMDE vorhanden ist.

Die Anwendung muss keine MQMD-Struktur bereitstellen, wenn eine gültige Nachrichtenkennung im Feld *MsgHandle* bereitgestellt wird. Wenn in diesem Feld nichts bereitgestellt wird, wird der Deskriptor der Nachricht vom Deskriptor der Nachrichtenkennungen übernommen.

Wenn die Anwendung eine Nachrichtenkennung statt einer MQMD-Struktur bereitstellt und MQGMO_PROPERTIES_FORCE_MQRFH2 angibt, schlägt der Aufruf mit Ursachencode MQRC_MD_ERROR fehl. Der Aufruf schlägt auch mit Ursachencode MQRC_MD_ERROR fehl, wenn die Anwendung keine MQMD-Struktur bereitstellt, MQGMO_PROPERTIES_AS_Q_DEF angibt und das Warteschlangenattribut *PropertyControl* MQPROP_FORCE_MQRFH2 ist.

Wenn Abgleichoptionen angegeben sind und der der Nachrichtenkennung zugeordnete Nachrichtendeskriptor verwendet wird, kommen die Eingabefelder, die zum Abgleichen verwendet werden, von der Nachrichtenkennung.

GetMsgOpts
Typ: MQGMO - Ein-/Ausgabe

Weitere Informationen finden Sie im Abschnitt „MQGMO – Nachrichtenabrufoptionen" auf Seite 348.

BufferLength
Typ: MQLONG - Eingabe

Dies ist die Länge des Bereichs *Buffer* in Byte. Geben Sie null für Nachrichten an, die keine Daten haben, oder wenn die Nachricht aus der Warteschlange entfernt und die Daten verworfen werden sollen (in diesem Fall müssen Sie MQGMO_ACCEPT_TRUNCATED_MSG angeben).

Anmerkung: Die Länge der längsten Nachricht, die aus der Warteschlange gelesen werden kann, wird durch das Warteschlangenattribut *MaxMsgLength* angegeben, siehe „Attribute für Warteschlangen" auf Seite 833.

Buffer
Typ: MQBYTExBufferLength - Ausgabe

Dies ist der Bereich zur Aufnahme der Nachrichtendaten. Richten Sie den Puffer an einem Grenzwert aus, der der Spezifik der Daten in der Nachricht entspricht. Die 4-Byte-Ausrichtung ist für die meisten Nachrichten geeignet (einschließlich Nachrichten mit IBM WebSphere MQ-Headerstrukturen); manche Nachrichten erfordern jedoch möglicherweise eine stringentere Ausrichtung. Eine Nachricht beispielsweise, die eine binäre 64-Bit-Ganzzahl enthält, erfordert eine 8-Byte-Ausrichtung.

Wenn *BufferLength* kleiner als die Nachrichtenlänge ist, wird der größtmögliche Teil der Nachricht in *Buffer* verschoben, wenn MQGMO_ACCEPT_TRUNCATED_MSG im Parameter *GetMsgOpts* angegeben ist (weitere Informationen finden Sie im Abschnitt MQGMO - Optionsfeld).

Der Zeichensatz und die Codierung der Daten in *Buffer* werden durch die Felder *CodedCharSetId* und *Encoding* angegeben, die vom Parameter *MsgDesc* zurückgegeben werden. Wenn diese Werte nicht den Werten entsprechen, die der Empfänger verlangt, muss dieser die Anwendungsnachrichtendaten in den erforderlichen Zeichensatz und in die erforderliche Codierung konvertieren. Die Option MQGMO_CONVERT kann (ggf. mit einem benutzerdefinierten Exit) verwendet werden, um die Nachrichtendaten zu konvertieren. Weitere Informationen zu dieser Option finden Sie im Abschnitt „MQGMO – Nachrichtenabrufoptionen” auf Seite 348.

Anmerkung: Alle anderen Parameter beim MQGET-Aufruf entsprechen dem Zeichensatz und der Codierung des lokalen Warteschlangenmanagers (durch das Warteschlangenmanagerattribut *CodedCharSetId* und MQENC_NATIVE angegeben).

Bei einem Fehlschlagen des Aufrufs kann sich der Pufferinhalt dennoch geändert haben.

In der Programmiersprache C wird der Parameter als Zeiger auf "void" deklariert: die Adresse jeden Datentyps kann als der Parameter angegeben werden.

Wenn der Parameter *BufferLength* null ist, wird *Buffer* nicht referenziert. Die Adresse, die von Programmen übergeben wird, die in C oder in System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

DataLength
Typ: MQLONG - Ausgabe

Dies gibt die Länge der Anwendungsdaten *in der Nachricht* in Byte an. Wenn dieser Wert größer als *BufferLength* ist, werden nur *BufferLength* Byte im Parameter *Buffer* zurückgegeben (d. h., die Nachricht wird gekürzt). Wenn der Wert null ist, enthält die Nachricht keine Anwendungsdaten.

Wenn *BufferLength* kleiner als die Nachrichtenlänge ist, wird *DataLength* dennoch vom Warteschlangenmanager aufgefüllt, wennMQGMO_ACCEPT_TRUNCATED_MSG beim Parameter *GetMsgOpts* angegeben ist (weitere Informationen finden Sie im Abschnitt MQGMO - Optionsfeld). Damit kann die Anwendung die Größe des Puffers bestimmen, die für die Aufnahme der Nachrichtendaten erforderlich ist, und dann den Aufruf mit einem Puffer entsprechender Größe erneut ausgeben.

Wenn allerdings die Option MQGMO_CONVERT angegeben ist und die konvertierten Nachrichtendaten zu lang für *Buffer* sind, ist der für *DataLength* zurückgegebene Wert:

- Bei vom Warteschlangenmanager definierten Formaten: Die Länge der *nicht konvertierten* Daten.

  Wenn die Daten aufgrund ihrer Spezifik während der Konvertierung expandieren, muss die Anwendung in diesem Fall einen Puffer zuordnen, der größer als der vom Warteschlangenmanager für *DataLength* zurückgegebene Wert ist.
- Bei anwendungsdefinierten Formaten der durch den Datenkonvertierungsexit zurückgegebene Wert.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Die aufgelisteten Ursachencodes kann der Warteschlangenmanager für den Parameter *Reason* zurückgeben. Wenn die Anwendung die Option MQGMO_CONVERT angibt und ein benutzerdefiniertes Exit aufgerufen wird, um einige oder die gesamten Nachrichtendaten zu konvertieren, entscheidet der Exit, welcher Wert für den Parameter *Reason* zurückgegeben wird. Daher sind andere als die dokumentierten Werte möglich.

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Die konvertierten Daten sind zu groß für den Puffer.

MQRC_CONVERTED_STRING_TOO_BIG
(2190, X'88E') Konvertierte Zeichenfolge zu groß für Feld

MQRC_DBCS_ERROR
(2150, X'866') DBCS-Zeichenfolge ungültig.

MQRC_FORMAT_ERROR
(2110, X'83E') Nachrichtenformat ungültig

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Nachrichtengruppe nicht vollständig

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Logische Nachricht nicht vollständig

MQRC_INCONSISTENT_CCSIDS
(2243, X'8C3') Nachrichtensegmente haben unterschiedliche CCSIDs

MQRC_INCONSISTENT_ENCODINGS
(2244, X'8C4') Nachrichtensegmente haben unterschiedliche Codierungen

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Ungültige Verwendung des Nachrichtentokens

MQRC_NO_MSG_LOCKED
(2209, X'8A1') Keine Nachricht gesperrt

MQRC_NOT_CONVERTED
(2119, X'847') Die Nachrichtendaten wurden nicht konvertiert.

MQRC_OPTIONS_CHANGED
(nnnn, X'xxx') Optionen, die konsistent sein müssen, wurden geändert

MQRC_PARTIALLY_CONVERTED
(2272, X'8E0') Nachrichtendaten teilweise konvertiert

MQRC_SIGNAL_REQUEST_ACCEPTED
(2070, X'816') Keine Nachrichtenrückgabe (aber Signalanforderung akzeptiert)

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') Quellenpufferparameter ungültig.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Die quellcodierte Zeichensatz-ID ist nicht gültig.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codierung gepackter Dezimalzahlen in der Nachricht wurde nicht erkannt.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codierung von Gleitkommazahlen in der Nachricht wurde nicht erkannt.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Quellcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Quellenlängenparameter ungültig.

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Zielpufferparameter ungültig.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') Die zielcodierte Zeichensatz-ID ist nicht gültig.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Durch Empfänger angegebene Ganzzahlcodierung nicht erkannt.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Durch Empfänger angegebene Gleitkommacodierung nicht erkannt.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Zielcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Abgeschnittene Nachricht zurückgegeben (Verarbeitung ist abgeschlossen).

MQRC_TRUNCATED_MSG_FAILED
(2080, X'820') Abgeschnittene Nachricht zurückgegeben (Verarbeitung nicht abgeschlossen)

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') Servicemodule für Datenkonvertierung können nicht geladen werden.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BACKED_OUT
(2003, X'7D3') Arbeitseinheit zurückgesetzt

MQRC_BUFFER_ERROR
(2004, X'7D4') Pufferparameter ungültig

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Listenüberschrift für Coupling-Facility-Struktur im Gebrauch

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Fehler bei Korrelations-ID.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Parameter Datenlänge ungültig.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') DB2-Subsystem nicht verfügbar

MQRC_GET_INHIBITED
(2016, X'7E0') wird für die Warteschlange unterdrückt.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Konflikt mit globaler Arbeitseinheit.

MQRC_GMO_ERROR
(2186, X'88A') Optionsstruktur zum Nachrichtenabruf ungültig.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Kennung für globale Arbeitseinheit belegt.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') Spezifikation zum Durchsuchen inkonsistent.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Nachricht unter Cursor nicht gültig für Abruf

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') Globale Arbeitseinheit im Konflikt mit lokaler Arbeitseinheit.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Abgleichoptionen ungültig

MQRC_MD_ERROR
(2026, X'7EA') Nachrichtendeskriptor ungültig

MQRC_MSG_ID_ERROR
(2206, X'89E') Fehler bei Nachrichten-ID

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Nachrichtenfolgenummer ungültig

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Verwendung des Nachrichtentokens ungültig.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Keine Nachricht verfügbar.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') Anzeigecursor nicht auf Nachricht positioniert.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Warteschlange nicht für Anzeige geöffnet

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Warteschlange nicht zur Eingabe geöffnet.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Warteschlange hat falschen Indextyp

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SECOND_MARK_NOT_ALLOWED
(2062, X'80E') Eine Nachricht ist bereits markiert

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Signal für diese Kennung ausstehend.

MQRC_SIGNAL1_ERROR
(2099, X'833') Signalfeld ungültig

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') In der aktuellen Arbeitseinheit können keine weiteren Nachrichten verarbeitet werden.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') Unterstützung für Synchronisationspunkt nicht verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Eintragung in globale Arbeitseinheit fehlgeschlagen.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') Mischung von Aufrufen für Arbeitseinheiten wird nicht unterstützt.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') Arbeitseinheit nicht zur Verwendung durch den Warteschlangenmanager verfügbar.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') Warteintervall in MQGMO ungültig

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Falsche MQGMO-Version bereitgestellt.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Falsche MQMD-Version bereitgestellt.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Eine abgerufene Nachricht wird normalerweise aus der Warteschlange gelöscht. Das Löschen erfolgt als Teil des MQGET-Aufrufs selbst oder als Teil eines Synchronisationspunkts.

   Es sind die Suchoptionen MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT und MQGMO_BROWSE_MSG_UNDER_CURSOR verfügbar.

2. Wird die Option MQGMO_LOCK mit einer der Suchoptionen angegeben, wird die gefundene Nachricht gesperrt, sodass sie nur für diese Kennung sichtbar ist.

   Wird die Option MQGMO_UNLOCK angegeben, wird eine zuvor gesperrte Nachricht freigegeben. In diesem Fall wird keine Nachricht abgerufen und die Parameter *MsgDesc*, *BufferLength*, *Buffer* und *DataLength* werden nicht überprüft oder geändert.

3. Bei Anwendungen, die einen MQGET-Aufruf ausgeben, kann die abgerufene Nachricht verloren gehen, wenn die Anwendung abnormal beendet oder die Verbindung während der Verarbeitung des Aufrufs unterbrochen wird. Dieses Problem tritt auf, weil der Stellvertreter, der auf der Plattform des Warteschlangenmanagers ausgeführt wird und den MQGET-Aufruf für die Anwendung ausgibt, den Verlust der Anwendung erst erkennen kann, wenn er die Nachricht an die Anwendung zurückgeben will, *nachdem* die Nachricht aus der Warteschlange entfernt wurde. Dieses Problem kann sowohl bei persistenten als auch bei nicht persistenten Nachrichten auftreten.

   Um das Risiko, Nachrichten auf diese Weise zu verlieren, zu eliminieren, sollten Sie Nachrichten immer in Arbeitseinheiten abrufen. Geben Sie dazu die Option MQGMO_SYNCPOINT im MQGET-Aufruf an und den MQCMIT- oder MQBACK-Aufruf, um die Arbeitseinheit nach Beendigung der Nachrichtenverarbeitung festzuschreiben bzw. zurückzusetzen. Wenn MQGMO_SYNCPOINT angegeben wird und der Client abnormal beendet oder die Verbindung getrennt wird, setzt der Stellvertreter die Arbeitseinheit auf dem Warteschlangenmanager zurück und die Nachricht wird in der Warteschlange wiederhergestellt. Weitere Informationen zu Synchronisationspunkten finden Sie im Abschnitt Überlegungen zum Synchronisationspunkt in WebSphere MQ-Anwendungen.

   Diese Situation kann bei IBM WebSphere MQ-Clients ebenso auftreten wie bei Anwendungen, die auf derselben Plattform wie der Warteschlangenmanager ausgeführt werden.

4. Wenn eine Anwendung eine Folge von Nachrichten in eine bestimmteWarteschlange in einer einzelnen Arbeitseinheit einreiht und diese Arbeitseinheit anschließend erfolgreich festschreibt, werden die Nachrichten wie folgt zum Abruf verfügbar:

   - Handelt es sich um eine *nicht gemeinsam genutzte* Warteschlange (d. h. eine lokale Warteschlange), werden alle Nachrichten innerhalb der Arbeitseinheit gleichzeitig verfügbar.
   - Wenn es sich bei der Warteschlange um eine *gemeinsam genutzte* Warteschlange handelt, werden Nachrichten innerhalb der Arbeitseinheit in der Reihenfolge verfügbar, in der sie eingereiht wurden, aber nicht alle gleichzeitig. Wenn das System stark ausgelastet ist, kann es vorkommen, dass die erste Nachricht in der Arbeitseinheit erfolgreich abgerufen wird, aber der MQGET-Aufruf für die zweite oder eine nachfolgende Nachricht in der Arbeitseinheit mit dem Ursachencode MQRC_NO_MSG_AVAILABLE fehlschlägt. In diesem Fall muss die Anwendung eine kurze Zeit warten und dann versuchen, die Operation zu wiederholen.

5. Wenn eine Anwendung eine Folge von Nachrichten in dieselbe Warteschlange einreiht, ohne Nachrichtengruppen zu verwenden,wird die Reihenfolge dieser Nachrichten beibehalten, sofern bestimm-

te Bedingungen erfüllt sind. Weitere Informationen finden Sie im Abschnitt Hinweise zur Verwendung von MQPUT. Sind die Bedingungen erfüllt, werden die Nachrichten der empfangenden Anwendung in der Reihenfolge angeboten, in der sie gesendet wurden. Dies gilt unter folgenden Voraussetzungen:

- Nur ein einziger Empfänger ruft Nachrichten aus der Warteschlange ab.

  Wenn mehrere Anwendungen Nachrichten aus der Warteschlange abrufen, müssen sie mit dem Absender den Mechanismus vereinbaren, mit dem Nachrichten, die zu einer Folge gehören, erkannt werden. Beispielsweise kann der Sender alle *CorrelId*-Felder in den Nachrichten einer Folge auf einen Wert setzen, der für die Nachrichtenfolge eindeutig ist.
- Der Empfänger ändert nicht willkürlich die Abrufreihenfolge, indem er beispielsweise eine bestimmte Nachrichten-ID (*MsgId*) oder Korrelations-ID (*CorrelId*) angibt.

Wenn die sendende Anwendung die Nachrichten als eine Nachrichtengruppe einreiht, werden die Nachrichten der empfangenden Anwendung in der richtigen Reihenfolge angeboten, wenn die empfangende Anwendung im MQGET-Aufruf die Option MQGMO_LOGICAL_ORDER angibt. Weitere Informationen zu Nachrichtengruppen finden Sie in den folgenden Abschnitten:

- MQMD - MsgFlags-Feld
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Wenn der Benutzer Nachrichten in einer Gruppe unter Synchronisationspunkt abruft, muss er sicherstellen, dass die gesamte Gruppe verarbeitet wird, bevor er versucht, die Transaktion zu beenden.

6. Anwendungen müssen überprüfen, ob das Feld *Feedback* des Parameters *MsgDesc* den Rückkopplungscode MQFB_QUIT enthält, und die Verarbeitung beenden, wenn dieser Wert gefunden wird. Weitere Informationen finden Sie im Abschnitt MQMD - Feedback-Feld.
7. Wenn die mit *Hobj* angegebene Warteschlange mit der Option MQOO_SAVE_ALL_CONTEXT geöffnet wurde und der Beendigungscode des MQGET-Aufrufs MQCC_OK oder MQCC_WARNING lautet, wird der Kontext, der der Warteschlangenkennung *Hobj* zugeordnet ist, auf den Kontext der Nachricht gesetzt, die abgerufen wurde (außer wenn die Option MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT oder MQGMO_BROWSE_MSG_UNDER_CURSOR angegeben ist, denn dann wird der Kontext als nicht verfügbar markiert).

   Sie können den gespeicherten Kontext in einem nachfolgenden MQPUT- oder MQPUT1-Aufruf speichern, indem Sie die Option MQPMO_PASS_IDENTITY_CONTEXT oder MQPMO_PASS_ALL_CONTEXT angeben. Dann kann der Kontext der empfangenen Nachricht als Ganzes oder zum Teil auf eine andere Nachricht übertragen werden (z. B. wenn die Nachricht an eine andere Warteschlange weitergeleitet wird). Weitere Informationen zum Nachrichtenkontext finden Sie im Abschnitt Nachrichtenkontext.
8. Wenn Sie die Option MQGMO_CONVERT im Parameter *GetMsgOpts* angeben, werden die Anwendungsnachrichtendaten in die von der empfangenden Anwendung angeforderte Darstellung konvertiert, bevor die Daten in den *Buffer* -Parameter gestellt werden:

   - Das Feld *Format* in den Steuerinformationen in der Nachricht gibt die Struktur der Anwendungsdaten an und die Felder *CodedCharSetId* und *Encoding* in den Steuerinformationen in der Nachricht geben ihre Zeichensatzkennung und Codierung an.
   - Die Anwendung, die den MQGET-Aufruf ausgibt, gibt in den Feldern *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* die Zeichensatzkennung und Codierung an, in die die Anwendungsnachrichtendaten konvertiert werden.

   Wenn die Konvertierung der Nachrichtendaten erforderlich ist, wird sie entweder vom Warteschlangenmanager selbst oder von einem benutzerdefinierten Exit ausgeführt, abhängig vom Wert des Felds *Format* in den Steuerinformationen in der Nachricht:

   - Die folgenden Formate werden vom Warteschlangenmanager konvertiert, sie werden als "integrierte" Formate bezeichnet:
     - MQFMT_ADMIN
     - MQFMT_CICS (nur z/OS)

  - MQFMT_COMMAND_1
  - MQFMT_COMMAND_2
  - MQFMT_DEAD_LETTER_HEADER
  - MQFMT_DIST_HEADER
  - MQFMT_EVENT Version 1
  - MQFMT_EVENT Version 2 (nurz/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 (nur z/OS)
  - MQFMT_XMIT_Q_HEADER
- Der Formatname MQFMT_NONE ist ein besonderer Wert, der bedeutet, dass das Format der Daten in der Nachricht nicht definiert ist. Deshalb versucht der Warteschlangenmanager in diesem Fall nicht, die Daten beim Abrufen der Nachricht aus der Warteschlange zu konvertieren.

  Anmerkung: Wenn MQGMO_CONVERT beim MQGET-Aufruf für eine Nachricht mit dem Formatnamen MQFMT_NONE angegeben wird und der Zeichensatz oder die Codierung der Nachricht von dem im Parameter *MsgDesc* angegebenen Zeichensatz oder der Codierung abweicht, wird die Nachricht im Parameter *Buffer* zurückgegeben (sofern keine anderen Fehler vorhanden sind), aber der Aufruf wird mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_FORMAT_ERROR abgeschlossen.

  Sie können MQFMT_NONE entweder dann verwenden, wenn die Nachrichtendaten so beschaffen sind, dass keine Konvertierung erforderlich ist, oder wenn die sendende und die empfangende Anwendung das Format, in dem die Nachrichtendaten gesendet werden sollen, miteinander vereinbart haben.
- Bei allen anderen Formaten wird die Nachricht an einen vom Benutzer geschriebenen Exit zur Konvertierung übergeben. Der Exit hat denselben Namen wie das Format, abgesehen von umgebungsspezifischen Zusätzen. Namen von benutzerspezifischen Formaten dürfen nicht mit den Buchstaben WebSphere MQ beginnen.

  Weitere Informationen zum Datenkonvertierungsexit finden Sie im Abschnitt „Datenkonvertierungsexit” auf Seite 908.

Benutzerdaten in der Nachricht können zwischen allen unterstützten Zeichensätzen und Codierungen konvertiert werden. Wenn die Nachricht eine oder mehrere WebSphere MQ-Headerstrukturen enthält, ist jedoch zu beachten, dass die Nachricht nicht aus einem oder in einen Zeichensatz mit Doppelbyte- oder Mehrfachbytezeichen für in Warteschlangennamen gültige Zeichen konvertiert werden kann. Wird dies versucht, führt dies zum Ursachencode MQRC_SOURCE_CCSID_ERROR oder MQRC_TARGET_CCSID_ERROR und die Nachricht wird unkonvertiert zurückgegeben. Der Unicode-Zeichensatz UCS-2 ist ein Beispiel für einen solchen Zeichensatz.

Bei der Rückgabe eines MQGET-Aufrufs zeigt folgender Ursachencode an, dass die Nachricht erfolgreich konvertiert wurde:

- MQRC_NONE

Der folgende Ursachencode gibt an, dass die Nachricht *möglicherweise* erfolgreich konvertiert wurde. Die Anwendung muss die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* überprüfen, um dies herauszufinden:

- MQRC_TRUNCATED_MSG_ACCEPTED

Alle anderen Ursachencodes bedeuten, dass die Nachricht nicht konvertiert wurde.

Anmerkung: Die Interpretation dieses Ursachencodes ist für Konvertierungen, die von einem vom Benutzer geschriebenen Exit durchgeführt werden, *nur* dann richtig, wenn der Exit den im Abschnitt „Datenkonvertierungsexit" auf Seite 908 beschriebenen Verarbeitungsleitlinien entspricht.

9. Bei Verwendung der objektorientierten Schnittstelle zum Abrufen von Nachrichten können Sie sich dafür entscheiden, keinen Puffer zur Aufnahme der Nachrichtendaten für einen MQGET-Aufruf anzugeben. Allerdings war es in früheren Versionen von WebSphere MQ möglich, dass MQGET mit Ursachencode MQRC_CONVERTED_MSG_TO_BIG fehlschlug, selbst wenn kein Puffer angegeben wurde. Wenn Sie in WebSphere MQ Version 7 eine Nachricht mit einer objektorientierten Anwendung abrufen, ohne die Größe des Nachrichtenempfangspuffers zu begrenzen, schlägt die Anwendung nicht mit Ursachencode MQRC_CONVERTED_MSG_TOO_BIG fehl, sondern empfängt die konvertierte Nachricht. Dies gilt für folgende Umgebungen:

   - .NET, einschließlich vollständig verwalteter Anwendungen
   - C + +
   - Java (WebSphere MQ Classes for Java)

   Anmerkung: Für alle Clients gilt, dass, wenn der Wert von *sharingConversations* null ist, der Kanal so funktioniert wie vor WebSphere MQ Version 7.0 und dass die Nachrichtenbehandlung auf das Verhalten von Version 6 zurückgesetzt wird. Wenn in einer solchen Situation der Puffer zu klein ist, um die konvertierte Nachricht aufzunehmen, wird die unkonvertierte Nachricht mit Ursachencode MQRC_CONVERTED_MSG_TOO_BIG zurückgegeben. Weitere Informationen zu *sharingConversations* finden Sie im Abschnitt Gemeinsamer Datenaustausch in einer Clientanwendung.

10. Bei den integrierten Formaten kann der Warteschlangenmanager eine *Standardkonvertierung* von Zeichenfolgen in der Nachricht durchführen, wenn die Option MQGMO_CONVERT angegeben wird. Für die Standardkonvertierung kann der Warteschlangenmanager einen installationsspezifischen Standardzeichensatz verwenden, der sich bei der Konvertierung von Zeichenfolgedaten dem tatsächlichen Zeichensatz annähert. Dies führt dazu, dass der MQGET-Aufruf mit Beendigungscode MQCC_OK fortgesetzt werden kann, statt mit Beendigungscode MQCC_WARNING und Ursachencode MQRC_SOURCE_CCSID_ERROR oder MQRC_TARGET_CCSID_ERROR beendet zu werden.

    Anmerkung: Die Verwendung eines angenäherten Zeichensatzes zur Konvertierung von Zeichenfolgedaten hat zur Folge, dass einige Zeichen möglicherweise nicht richtig konvertiert werden. Um dies zu verhindern, sollten in der Zeichenfolge Zeichen verwendet werden, die sowohl im tatsächlichen Zeichensatz als auch im Standardzeichensatz vorkommen.

    Die Standardkonvertierung wird sowohl auf Anwendungsnachrichtendaten als auch auf Zeichenfelder in den MQMD- und MQMDE-Strukturen angewendet:

    - Die Standardkonvertierung von Anwendungsnachrichtendaten findet nur statt, wenn *alle* folgenden Bedingungen zutreffen:
      - Die Anwendung gibt die Option MQGMO_CONVERT an.
      - Die Nachricht enthält Daten, die entweder aus einem oder in einen Zeichensatz konvertiert werden müssen, der nicht unterstützt wird.
      - Die Standardkonvertierung wurde bei der Installation oder beim Neustart des Warteschlangenmanagers aktiviert.
    - Die Standardkonvertierung der Zeichenfelder in den MQMD- und MQMDE-Strukturen findet bei Bedarf statt, sofern die Standardkonvertierung für den Warteschlangenmanager aktiviert ist. Die Konvertierung wird auch dann durchgeführt, wenn die Anwendung die Option MQGMO_CONVERT nicht im MQGET-Aufruf angegeben hat.

11. Für die Programmiersprache Visual Basic gilt Folgendes:

- Wenn die Größe des Parameters *Buffer* kleiner als die Länge ist, die im Parameter *Buffer-Length* angegeben ist, schlägt der Aufruf mit Ursachencode MQRC_STORAGE_NOT_AVAILABLE fehl.
- Der Parameter *Buffer* wird als Typ `String` deklariert. Wenn es sich bei den Daten, die aus der Warteschlange abgerufen werden sollen, nicht um Daten des Typs `String` handelt, verwenden Sie denMQGETAny-Aufruf anstelle des MQGET-Aufrufs.

  Der MQGETAny-Aufruf hat dieselben Parameter wie der MQGET-Aufruf, außer dass der Parameter *Buffer* als Typ Any deklariert wird, sodass jeder beliebige Datentyp abgerufen werden kann. Dies bedeutet jedoch, dass der Parameter *Buffer* nicht daraufhin überprüft werden kann, ob er mindestens eine Größe von *BufferLength* Bytes aufweist.

12. Wenn Vorauslesen aktiviert ist, werden nicht alle MQGET-Optionen unterstützt. Die folgende Tabelle zeigt, welche Optionen zulässig sind und ob sie zwischen MQGET-Aufrufen ausgetauscht werden können.

Tabelle 566. Zulässige MQGET-Optionen bei aktiviertem Vorauslesen

	Bei aktiviertem Vorauslesen zulässig und kann zwischen MQGET-Aufrufen ausgetauscht werden	**Bei aktiviertem Vorauslesen zulässig, kann aber nicht zwischen MQGET-Aufrufen ausgetauscht werden[a]**	**MQGET-Optionen, die bei aktiviertem Vorauslesen nicht zulässig sind[b]**
MQGET MD-Werte	MsgId[c] CorrelId[c]	Encoding CodedCharSetId	
MQGET MQGMO-Optionen	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
MQGMO-Werte		MsgHandle	

a. Wenn diese Optionen zwischen MQGET-Aufrufen ausgetauscht werden, wird der Ursachencode MQRC_OPTIONS_CHANGED zurückgegeben.

b. Wenn diese Optionen im ersten MQGET-Aufruf angegeben werden, wird das Vorauslesen inaktiviert. Werden diese Optionen in einem nachfolgenden MQGET-Aufruf angegeben, wird Ursachencode MQRC_OPTIONS_ERROR zurückgegeben.

c. Clientanwendungen müssen erkennen können, dass bei einer Änderung der Werte für 'MsgId' und 'CorrelId' zwischen MQGET-Aufrufen Nachrichten mit den früheren Werten möglicherweise bereits an den Client gesendet wurden und im Vorauslesepuffer des Clients verbleiben, bis sie verarbeitet (oder automatisch gelöscht) werden.

d. Der erste MQGET-Aufruf bestimmt, ob Nachrichten aus einer Warteschlange angezeigt oder abgerufen werden, wenn Vorauslesen aktiviert ist. Wenn die Anwendung versucht, Anzeige und Abruf zu kombinieren, wird Ursachencode MQRC_OPTIONS_CHANGED zurückgegeben.

e. MQGMO_MSG_UNDER_CURSOR ist bei aktiviertem Vorauslesen nicht möglich. Nachrichten können angezeigt oder abgerufen werden, wenn Vorauslesen aktiviert ist, aber eine Kombination der beiden Funktionen ist nicht möglich.

13. Anwendungen können nicht festgeschriebene Nachrichten nur dann beim Abruf löschen, wenn diese Nachrichten in derselben lokalen Arbeitseinheit eingereiht wurden, in der sie abgerufen werden. Anwendungen können nicht festgeschriebene Nachrichten nicht abrufen, ohne sie beim Abruf zu löschen.
14. Nachrichten unter einem Anzeigecursor können in einer Arbeitseinheit abgerufen werden. Es nicht möglich, eine nicht festgeschriebene Nachricht auf diese Weise abzurufen.

C-Aufruf

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

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

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

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
        CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
              BUFFER,DATALENGTH,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Objektattribute abfragen

Der Aufruf MQINQ gibt eine Ganzzahlenfeldgruppe und eine Zeichenfolgegruppe mit den Attributen eines Objekts zurück.

Folgende Objekttypen sind gültig:

- Warteschlangenmanager
- Warteschlange
- Namensliste
- Prozessdefinition

Syntax

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem vorherigen MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Für CICS-Anwendungen unter z/OS und Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, kann der Aufruf MQCONN übergangen und folgender Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Hobj
Typ: MQHOBJ - Eingabe

Diese Kennung steht für das Objekt (eines beliebigen Typs) mit erforderlichen Attributen. Die Kennung muss von einem vorherigen MQOPEN-Aufruf, in dem die Option MQOO_INQUIRE angegeben war, zurückgegeben werden.

SelectorCount
Typ: MQLONG - Eingabe

Dies ist der Zähler für Selektoren, die im Array *Selectors* angegeben werden. Er gibt die Anzahl der Attribute an, die zurückgegeben werden müssen. Null ist ein gültiger Wert. Die maximal zulässige Anzahl ist 256.

Selectors
Typ: MQLONG × *SelectorCount* - Eingabe

Dies ist eine Feldgruppe aus *SelectorCount* Attributselektoren; jeder Selektor gibt ein Attribut (Ganzzahl oder Zeichen) mit einem Wert an, der erforderlich ist.

Jeder Selektor muss für den Typ des in *Hobj* angegebenen Objekts gültig sein; andernfalls schlägt der Aufruf mit Beendigungscode MQCC_FAILED und Ursachencode MQRC_SELECTOR_ERROR fehl.

Sonderfall Warteschlangen:

- Wenn der Selektor für Warteschlangen eines beliebigen Typs ungültig ist, schlägt der Aufruf mit Beendigungscode MQCC_FAILED und Ursachencode MQRC_SELECTOR_ERROR fehl.
- Wenn der Selektor nur für Warteschlangen anderer Typen als den Typ des Objekts gilt, wird der Aufruf mit Beendigungscode MQCC_WARNING und Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE erfolgreich beendet.
- Wenn die abgefragte Warteschlange eine Clusterwarteschlange ist, hängt die Frage, welche Selektoren gültig sind, davon ab, wie die Warteschlange aufgelöst wurde (weitere Informationen siehe Abschnitt „Hinweise zur Verwendung" auf Seite 715).

Die Selektoren können in beliebiger Reihenfolge angegeben werden. Attributwerte für Ganzzahlenattributselektoren (MQIA_*-Selektoren) werden im Parameter *IntAttrs* in derselben Reihenfolge zurückgegeben, in der diese Selektoren im Parameter *Selectors* angegeben sind. Attributwerte für Zeichenattributselektoren (MQCA_*-Selektoren) werden im Parameter *CharAttrs* in derselben Reihenfolge zurückgegeben, in der diese Selektoren angegeben sind. MQIA_*-Selektoren können mit MQCA_*-Selektoren verzahnt werden. Wichtig ist allein die relative Reihenfolge innerhalb der einzelnen Typen.

Anmerkung:

1. Die Ganzzahlen- und Zeichenattributselektoren werden in zwei verschiedenen Bereichen angegeben. Die MQIA_*-Selektoren befinden sich im Bereich MQIA_FIRST bis MQIA_LAST und die MQCA_*-Selektoren im Bereich MQCA_FIRST bis MQCA_LAST.

   In jedem Bereich legen die Konstanten MQIA_LAST_USED und MQCA_LAST_USED den höchsten Wert fest, der von den Warteschlangenmanagern akzeptiert wird.
2. Wenn alle MQIA_*-Selektoren zuerst genannt werden, können zur Adressierung entsprechender Elemente in den Feldgruppen *Selectors* und *IntAttrs* dieselben Elementnummern verwendet werden.
3. Wenn der Parameter *SelectorCount* auf null gesetzt ist, wird *Selectors* nicht referenziert. In diesem Fall kann die Parameteradresse null sein, die von Programmen übergeben wird, die in C oder S/390-Assembler geschrieben sind.

Die Attribute, die abgefragt werden können, sind in den folgenden Tabellen aufgelistet. Für die MQCA_*-Selektoren wird die Konstante, die die Länge der Ergebniszeichenfolge in *CharAttrs* in Byte festlegt, in Klammern angegeben.

In den folgenden Tabellen sind die Selektoren wie folgt in alphabetischer Reihenfolge nach Objekt aufgelistet:

- Tabelle 567 auf Seite 702 MQINQ-Attributselektoren für Warteschlangen
- Tabelle 568 auf Seite 705 MQINQ-Attributselektoren für Namenslisten
- Tabelle 569 auf Seite 705 MQINQ-Attributselektoren für Prozessdefinitionen
- Tabelle 570 auf Seite 705 MQINQ-Attributselektoren für den Warteschlangenmanager

Alle Selektoren werden auf allen IBM WebSphere MQ-Plattformen unterstützt, sofern in der Spalte **Anmerkung** nicht Folgendes angegeben ist:

Nicht z/OS
Wird auf allen Plattformen **außer** z/OS unterstützt

z/OS
Wird **nur** unter z/OS unterstützt

Tabelle 567. MQINQ-Attributselektoren für Warteschlangen

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Datum der letzten Änderung	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Uhrzeit der letzten Änderung	
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Name der Warteschlange zum Wiedereinreihen überzähliger zurückgesetzter Nachrichten	
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Name der Warteschlange, in den der Alias aufgelöst wird	
MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Name der Coupling-Facility-Struktur	z/OS
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Name des Clustersenderkanals, der diese Warteschlange als Übertragungswarteschlange verwendet.	nichtz/OS
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Clustername	
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Clusternamensliste	
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Erstellungsdatum der Warteschlange	
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Erstellungsuhrzeit der Warteschlange	
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Name der Initialisierungswarteschlange	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Name der Prozessdefinition	
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Warteschlangenbeschreibung	
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Warteschlangenname	

Tabelle 567. MQINQ-Attributselektoren für Warteschlangen (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Name des fernen Warteschlangenmanagers	
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Name der im fernen Warteschlangenmanager bekannten fernen Warteschlange	
MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Name der Speicherklasse	z/OS
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Daten des Auslösers	
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Name der Übertragungswarteschlange	
MQIA_ACCOUNTING_Q	MQLONG	Steuert die Erfassung von Abrechnungsdaten für Warteschlange	Nicht z/OS
MQIA_BACKOUT_THRESHOLD	MQLONG	Rücksetzschwellenwert	
MQIA_CLWL_Q_PRIORITY	MQLONG	Priorität der Warteschlange	
MQIA_CLWL_Q_RANK	MQLONG	Rang der Warteschlange	
MQIA_CLWL_USEQ	MQLONG	Ferne Warteschlangen verwenden	
MQIA_CURRENT_Q_DEPTH	MQLONG	Anzahl an Nachrichten in Warteschlange	
MQIA_DEF_BIND	MQLONG	Standardbindung	
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Standardoption für Öffnen für Eingaben	
MQIA_DEF_PERSISTENCE	MQLONG	Standardpersistenz für Nachrichten	
MQIA_DEF_PRIORITY	MQLONG	Standardpriorität für Nachr.	
MQIA_DEFINITION_TYPE	MQLONG	Warteschlangendefinitionstyp	
MQIA_DIST_LISTS	MQLONG	Unterstützung Verteilerliste	Nicht z/OS
MQIA_HARDEN_GET_BACKOUT	MQLONG	Gibt an, ob Rücksetzungszähler permanent gespeichert werden soll	
MQIA_INDEX_TYPE	MQLONG	Typ des für Warteschlange verwalteten Indexes	z/OS
MQIA_INHIBIT_GET	MQLONG	Gibt an, ob GET-Operationen zulässig sind	
MQIA_INHIBIT_PUT	MQLONG	Gibt an, ob PUT-Operationen zulässig sind	
MQIA_MAX_MSG_LENGTH	MQLONG	Maximale Nachrichtenlänge	
MQIA_MAX_Q_DEPTH	MQLONG	Maximal zulässige Anzahl an Nachrichten in Warteschlange	
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Gibt an, ob Nachrichtenpriorität relevant ist	

Tabelle 567. MQINQ-Attributselektoren für Warteschlangen (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQIA_NPM_CLASS	MQLONG	Zuverlässigkeitsstufe für nicht persistente Nachrichten	
MQIA_OPEN_INPUT_COUNT	MQLONG	Anzahl MQOPEN-Aufrufe, die die Warteschlange für Eingaben geöffnet haben	
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Anzahl MQOPEN-Aufrufe, die die Warteschlange für Ausgaben geöffnet haben	
MQIA_PROPERTY_CONTROL	MQLONG	Eigenschaftssteuerattribut	
MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Steuerattribut für 'Warteschlangenlänge hoch'-Ereignisse	Nicht z/OS
MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Oberer Grenzwert für Warteschlangenlänge	Nicht z/OS
MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Steuerattribut für 'Warteschlangenlänge niedrig'-Ereignisse	Nicht z/OS
MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Unterer Grenzwert für Warteschlangenlänge	Nicht z/OS
MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Steuerattribut für Ereignisse vom Typ "Queue Depth Max" (Warteschlangenlänge maximal).	Nicht z/OS
MQIA_Q_SERVICE_INTERVAL	MQLONG	Grenzwert für Warteschlangenserviceintervall	Nicht z/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Steuerattribut für Warteschlangenserviceintervall-Ereignisse	Nicht z/OS
MQIA_Q_TYPE	MQLONG	Warteschlangentyp	
MQIA_QSG_DISP	MQLONG	Disposition der Gruppe mit gemeinsamer Warteschlange	z/OS
MQIA_RETENTION_INTERVAL	MQLONG	Warteschlangensicherungsintervall	
MQIA_SCOPE	MQLONG	Warteschlangendefinitionsbereich	Nicht z/OS
MQIA_SHAREABILITY	MQLONG	Gibt an, ob Warteschlange gemeinsam für Eingaben genutzt werden kann	
MQIA_STATISTICS_Q	MQLONG	Steuert die Erfassung von Statistikdaten für Warteschlange	Nicht z/OS
MQIA_TRIGGER_CONTROL	MQLONG	Auslösesteuerung	
MQIA_TRIGGER_DEPTH	MQLONG	Auslösertiefe	
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Schwellenwertnachrichtenpriorität für Auslöser	
MQIA_TRIGGER_TYPE	MQLONG	Auslösertyp	
MQIA_USAGE	MQLONG	Verwendung	

Tabelle 568. MQINQ-Attributselektoren für Namenslisten

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Datum der letzten Änderung	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Uhrzeit der letzten Änderung	
MQCA_NAMELIST_DESC	MQ_NAME-LIST_DESC_LENGTH	Namenslistenbeschreibung	
MQCA_NAMELIST_NAME	MQ_NAME-LIST_NAME_LENGTH	Name des Namenslistenobjekts	
MQIA_NAMELIST_TYPE	MQLONG	Namenslistentyp	z/OS
MQCA_NAMES	MQ_Q_NAME_LENGTH × Anzahl der Namen in der Liste	Namen in der Namensliste	
MQIA_NAME_COUNT	MQLONG	Anzahl Namen in der Namensliste	
MQIA_QSG_DISP	MQLONG	Disposition der Gruppe mit gemeinsamer Warteschlange	z/OS

Tabelle 569. MQINQ-Attributselektoren für Prozessdefinitionen

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Datum der letzten Änderung	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Uhrzeit der letzten Änderung	
MQCA_APPL_ID	MQ_PRO-CESS_APPL_ID_LENGTH	Anwendungs-ID	
MQCA_ENV_DATA	MQ_PROCESS_ENV_DA-TA_LENGTH	Umgebungsdaten	
MQCA_PROCESS_DESC	MQ_PRO-CESS_DESC_LENGTH	Beschreibung der Prozessdefinition	
MQCA_PROCESS_NAME	MQ_PRO-CESS_NAME_LENGTH	Name der Prozessdefinition	
MQCA_USER_DATA	MQ_PROCESS_USER_DA-TA_LENGTH	Benutzerdaten	
MQIA_APPL_TYPE	MQLONG	Anwendungstyp	
MQIA_QSG_DISP	MQLONG	Disposition der Gruppe mit gemeinsamer Warteschlange	z/OS

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Datum der letzten Änderung	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Uhrzeit der letzten Änderung	

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Name des automatischen Kanaldefinitionsexits	
MQCA_CHINIT_SERVICE_PARM		Reserviert für Verwendung durch IBM	
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	An Exit für Clusterauslastung übergebene Daten	
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Name des Exits für Clusterauslastung	
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Name der Eingabewarteschlange für Systembefehle	
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Name der Warteschlange für nicht zustellbare Nachrichten	
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Name der standardmäßigen Übertragungswarteschlange	
MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Name der Gruppe für die TCP-Empfangsfunktion, die eingehende Übertragungen für die teilzunehmende Gruppe mit gemeinsamer Warteschlange bearbeitet. Der Name wird angewendet, wenn Workload Manager Dynamic Domain Name Services verwendet werden.	z/OS
MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Benutzer-ID der gruppeninternen Warteschlangensteuerung	z/OS
MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Beschreibung der zugehörigen Installation	Nicht z/OS . nicht IBM i
MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Name der Installation, die dem Warteschlangenmanager zugeordnet ist	Nicht z/OS . nicht IBM i
MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Pfad, in dem das zugehörige IBM WebSphere MQ installiert ist	Nicht z/OS . nicht IBM i
MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Generischer LU-Name für das LU 6.2-Empfangsprogramm, das eingehende Übertragungen für die zu verwendende Gruppe mit gemeinsamer Warteschlange verarbeitet	z/OS

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Name der für ausgehende LU 6.2-Übertragungen zu verwendenden LU. Setzen Sie diesen Namen auf dieselbe LU, die das Empfangsprogramm für eingehende Übertragungen verwendet.	z/OS
MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Suffix des SYS1.PARMLIB-Mitglieds APPCPM*xx*, das die LUADD für diesen Kanalinitiator benennt.	z/OS
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Name eines hierarchisch verbundenen Warteschlangenmanagers, der als diesem Warteschlangenmanager übergeordnet benannt ist	
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Beschreibung des Warteschlangenmanagers	
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTI-FIER_LENGT	Warteschlangenmanager-I (H)	
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Name des lokalen Warteschlangenmanagers	
MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Name der Gruppe mit gemeinsamer Warteschlange	z/OS
MQCA_REPOSITORY_NAME	MQ_CLUS-TER_NAME_LENGTH	Name des Clusters, für das der Warteschlangenmanager Repository-Services bereitstellt	
MQCA_REPOSITORY_NAMELIST	MQ_NAME-LIST_NAME_LENGTH	Name des Namenslistenobjekts mit den Namen von Clustern, für die der Warteschlangenmanager Repository-Services bereitstellt	
MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Name des verwendeten TCP/IP-Systems	z/OS
MQIA_ACCOUN-TING_CONN_OVERRIDE	MQLONG	Abrechnungseinstellungen überschreiben	Nicht z/OS
MQIA_ACCOUNTING_INTERVAL	MQLONG	Gibt an, wie oft temporäre Abrechnungssätze geschrieben werden sollen	Nicht z/OS
MQIA_ACCOUNTING_MQI	MQLONG	Steuert die Erfassung von Abrechnungsinformationen für MQI-Daten	Nicht z/OS
MQIA_ACCOUNTING_Q	MQLONG	Steuert die Erfassung von Abrechnungsinformationen für Warteschlangen	Nicht z/OS
MQIA_ACTIVE_CHANNELS	MQLONG	Maximale Anzahl Kanäle, die zu jeder Zeit aktiv sein können	z/OS

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQIA_ADOPTNEWMCA_CHECK	MQLONG	Überprüfte Elemente, um zu ermitteln, ob ein Nachrichtenkanalagent angenommen wird. Die Überprüfung wird durchgeführt, wenn ein neuer eingehender Kanal erkannt wird, der denselben Namen wie ein bereits aktiver Nachrichtenkanalagent hat.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Zeit in Sekunden, die ein neuer Kanal wartet, bis der verwaiste Kanal geschlossen wird	Nicht z/OS
MQIA_ADOPTNEWMCA_TYPE	MQLONG	Gibt an, ob eine verwaiste Instanz eines Nachrichtenkanalagenten eines bestimmten Kanaltyps automatisch erneut gestartet wird, wenn eine neue eingehende Kanalanforderung, die den AdoptNewMCACheck-Parametern entspricht, erkannt wird	z/OS
MQIA_AUTHORITY_EVENT	MQLONG	Steuerattribut für Berechtigungsereignisse	Nicht z/OS
MQIA_BRIDGE_EVENT	MQLONG	Steuerattribut für IMS-Brückenereignisse	z/OS
MQIA_CHANNEL_AUTO_DEF	MQLONG	Steuerattribut für automatische Kanaldefinition	Nicht z/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Steuerattribut für Ereignisse der automatischen Kanaldefinition	Nicht z/OS
MQIA_CHANNEL_EVENT	MQLONG	Steuerattribut für Kanalereignisse	
MQIA_CHINIT_ADAPTERS	MQLONG	Anzahl zu verwendender Adapter-Subtasks für Verarbeitung von IBM WebSphere MQ-Aufrufen	z/OS
MQIA_CHINIT_DISPATCHERS	MQLONG	Anzahl zu verwendender Dispatcher für den Kanalinitiator	z/OS
MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Gibt an, ob der Kanalinitiatortrace automatisch gestartet werden soll	z/OS
MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Größe des Tracedatenspeicherbereichs des Kanalinitiators (in MB)	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Länge der Clusterauslastung.	
MQIA_CLWL_MRU_CHANNELS	MQLONG	Anzahl der zuletzt verwendeten Kanäle für Clusterlastausgleich	
MQIA_CLWL_USEQ	MQLONG	Ferne Warteschlangen verwenden	
MQIA_CODED_CHAR_SET_ID	MQLONG	ID des codierten Zeichensatzes	
MQIA_COMMAND_EVENT	MQLONG	Steuerattribut für Befehlsereignisse	

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQIA_COMMAND_LEVEL	MQLONG	Vom Warteschlangenmanager unterstützte Befehlsebene	
MQIA_CONFIGURATION_EVENT	MQLONG	Steuerattribut für Konfigurationsereignisse	Nicht z/OS
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Standardmäßig zu verwendender Übertragungswarteschlangentyp für Clustersenderkanäle.	nicht z/OS
MQIA_DIST_LISTS	MQLONG	Unterstützung Verteilerliste	Nicht z/OS
MQIA_DNS_WLM	MQLONG	Gibt an, ob das TCP-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet, mit Workload Manager for Dynamic Domain Name Services registriert wird.	z/OS
MQIA_EXPIRY_INTERVAL	MQLONG	Intervall zwischen Überprüfungen auf abgelaufene Nachrichten	z/OS
MQIA_GROUP_UR	MQLONG	Steuerattribut dafür, ob GROUP-Arbeitseinheiten mit Wiederherstellung für diesen Warteschlangenmanager aktiviert sind. Die Disposition der GROUP-Arbeitseinheit mit Wiederherstellung ist nur verfügbar, wenn der Warteschlangenmanager Mitglied einer Gruppe mit gemeinsamer Warteschlange ist.	z/OS
MQIA_IGQ_PUT_AUTHORITY	MQLONG	PUT-Berechtigung für gruppeninterne Warteschlangensteuerung	z/OS
MQIA_INHIBIT_EVENT	MQLONG	Steuerattribut für Sperrereignisse	Nicht z/OS
MQIA_INTRA_GROUP_QUEUING	MQLONG	Unterstützung für gruppeninterne Warteschlangensteuerung	z/OS
MQIA_LISTENER_TIMER	MQLONG	Zeitintervall (in Sekunden) zwischen Versuchen von IBM WebSphere MQ zum Neustart des Empfangsprogramms, wenn APPC oder TCP/IP fehlgeschlagen ist	z/OS
MQIA_LOCAL_EVENT	MQLONG	Steuerattribut für lokale Ereignisse	Nicht z/OS
MQIA_LOGGER_EVENT	MQLONG	Steuerattribut für Sperrereignisse	Nicht z/OS
MQIA_LU62_CHANNELS	MQLONG	Maximale Anzahl Kanäle, die aktiv sein können, oder Clients, die über das LU 6.2-Übertragungsprotokoll verbunden sein können	z/OS

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Zeitintervall (in Millisekunden), nach dem der Warteschlangenmanager eine Markierung automatisch aus Anzeigenachrichten entfernen kann.  **Achtung:** Dieser Wert sollte nicht niedriger als der Standardwert 5000 festgelegt werden.	
MQIA_MAX_CHANNELS	MQLONG	Maximale Anzahl von Kanälen, die aktiv sein können (einschließlich Serververbindungskanäle mit verbundenen Clients)	z/OS
MQIA_MAX_HANDLES	MQLONG	Maximale Anzahl von Kennungen	
MQIA_MAX_MSG_LENGTH	MQLONG	Maximale Nachrichtenlänge	
MQIA_MAX_PRIORITY	MQLONG	Maximale Priorität	
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Maximale Anzahl nicht festgeschriebener Nachrichten in einer Arbeitseinheit	
MQIA_OUTBOUND_PORT_MAX	MQLONG	Legt zusammen mit MQIA_OUTBOUND_PORT_MIN den Bereich der Portnummern fest, die bei der Bindung abgehender Kanäle verwendet werden sollen	z/OS
MQIA_OUTBOUND_PORT_MIN	MQLONG	Legt zusammen mit MQIA_OUTBOUND_PORT_MAX den Bereich der Portnummern fest, die bei der Bindung abgehender Kanäle verwendet werden sollen	z/OS
MQIA_PERFORMANCE_EVENT	MQLONG	Steuerattribut für Leistungsereignisse	Nicht z/OS
MQIA_PLATFORM	MQLONG	Plattform, auf der sich der Warteschlangenmanager befindet	
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Gibt an, ob Sicherheitsfunktionen von WebSphere MQ Advanced Message Security für einen Warteschlangenmanager verfügbar sind.	
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	Die Anzahl der Versuche, eine fehlgeschlagene Befehlsnachricht unter einem Synchronisationspunkt erneut zu verarbeiten	

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)

Selektor	Feldlänge	Beschreibung	Hinweis
MQIA_PUBSUB_MODE	MQLONG	Steuert, ob die Publish/Subscribe-Engine und die eingereihte Publish/Subscribe-Schnittstelle aktiv sind. Publish/Subscribe-Anwendungen, die die Anwendungsprogrammierschnittstelle verwenden, benötigen die Publish/Subscribe-Engine. Für Warteschlangen, die von der eingereihten Publish/Subscribe-Schnittstelle überwacht werden, muss die Publish/Subscribe-Schnittstelle aktiv sein.	
MQIA_PUBSUB_NP_MSG	MQLONG	Gibt an, ob eine nicht zugestellte Nachricht gelöscht (oder beibehalten) wird	
MQIA_PUBSUB_NP_RESP	MQLONG	Steuert das Verhalten von nicht zugestellten Antwortnachrichten	
MQIA_PUBSUB_SYNC_PT	MQLONG	Gibt an, ob nur persistente (oder alle) Nachrichten unter Synchronisationspunkt verarbeitet werden sollen	
MQIA_QMGR_CFCONLOS	MQLONG	Gibt die Aktion an, die ausgeführt werden soll, wenn CFCONLOS auf ASQMGR gesetzt ist und der Warteschlangenmanager die Verbindung mit der Verwaltungsstruktur oder einer CF-Struktur verliert.	z/OS
MQIA_RECEIVE_TIMEOUT	MQLONG	Gibt an, wie lange ein TCP/IP-Kanal ungefähr auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt. Dies ist der numerische Wert, der mit MQIA_RECEIVE_TIMEOUT_TYPE festgelegt wird.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Mindestzeit, die ein TCP/IP-Kanal auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	Gibt an, wie lange ein TCP/IP-Kanal ungefähr auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt. MQIA_RECEIVE_TIMEOUT_TYPE ist das für MQIA_RECEIVE_TIMEOUT geltende Qualifikationsmerkmal.	z/OS
MQIA_REMOTE_EVENT	MQLONG	Steuerattribut für ferne Ereignisse	Nicht z/OS

Tabelle 570. MQINQ-Attributselektoren für den Warteschlangenmanager (Forts.)			
Selektor	**Feldlänge**	**Beschreibung**	**Hinweis**
MQIA_SECURITY_CASE	MQLONG	Groß-/Kleinschreibung von Sicherheitsprofilen	z/OS
MQIA_SSL_EVENT	MQLONG	Steuerattribut für Kanalereignisse	
MQIA_SSL_FIPS_REQUIRED	MQLONG	Nur FIPS-zertifizierte Algorithmen für Verschlüsselung verwenden	
MQIA_SSL_RESET_COUNT	MQLONG	Rückstellzähl. f. SSL-Schlüss.	
MQIA_START_STOP_EVENT	MQLONG	Steuerattribut für Start-/Stoppereignisse	Nicht z/OS
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Steuert die Erfassung von statistischen Überwachungsdaten für Clustersenderkanäle	Nicht z/OS
MQIA_STATISTICS_CHANNEL	MQLONG	Steuert die Erfassung von statistischen Daten für Kanäle	Nicht z/OS
MQIA_STATISTICS_INTERVAL	MQLONG	Gibt an, wie oft statistische Überwachungsdaten geschrieben werden sollen	Nicht z/OS
MQIA_STATISTICS_MQI	MQLONG	Steuert die Erfassung von statistischen Überwachungsdaten für Warteschlangenmanager	Nicht z/OS
MQIA_STATISTICS_Q	MQLONG	Steuert die Erfassung von Statistikdaten für Warteschlangen	Nicht z/OS
MQIA_SYNCPOINT	MQLONG	Synchronisationspunktverfügbarkeit	
MQIA_TCP_CHANNELS	MQLONG	Maximale Anzahl von Kanälen, die aktiv sein können, oder von Clients, die über das TCP/IP-Übertragungsprotokoll verbunden sein können	z/OS
MQIA_TCP_KEEP_ALIVE	MQLONG	Gibt an, ob mithilfe der TCP-Funktion KEEPALIVE überprüft werden soll, ob die Gegenseite der Verbindung noch verfügbar ist	z/OS
MQIA_TCP_STACK_TYPE	MQLONG	Gibt an, ob der Kanalinitiator nur den in TCPNAME angegebenen TCP/IP-Adressraum verwenden oder optional eine Bindung zu einer beliebigen ausgewählten TCP/IP-Adresse herstellen kann	z/OS
MQIA_TRACE_ROUTE_RECORDING	MQLONG	Steuert die Aufzeichnung von Trace-Route-Informationen	z/OS
MQIA_TREE_LIFE_TIME	MQLONG	Lebensdauer von nicht genutzten, verwaltungsfremden Themen	
MQIA_TRIGGER_INTERVAL	MQLONG	Auslöseintervall	

IntAttrCount
Typ: MQLONG - Eingabe

Dies ist die Anzahl der Elemente in der Feldgruppe *IntAttrs*. Null ist ein gültiger Wert.

Wenn IntAttrCount mindestens der Anzahl von MQIA_*-Selektoren im Parameter *Selectors* entspricht, werden alle angeforderten Ganzzahlenattribute zurückgegeben.

IntAttrs
Typ: MQLONG ×*IntAttrCount* - Ausgabe

Dies ist ein Array aus *IntAttrCount* Ganzzahlattributwerten.

Ganzzahlenattributwerte werden in derselben Reihenfolge wie die MQIA_*-Selektoren im Parameter *Selectors* zurückgegeben. Wenn die Feldgruppe mehr Elemente enthält, als es MQIA_*-Selektoren gibt, bleiben die überzähligen Elemente unverändert.

Wenn *Hobj* eine Warteschlange angibt, aber der Attributselektor für den zugehörigen Warteschlangentyp ungültig ist, wird der Sonderwert MQIAV_NOT_APPLICABLE zurückgegeben. Er wird für das entsprechende Element im Array *IntAttrs* zurückgegeben.

Wenn der Parameter *IntAttrCount* oder *SelectorCount* auf null gesetzt ist, wird *IntAttrs* nicht referenziert. In diesem Fall kann die Parameteradresse null sein, die von Programmen übergeben wird, die in C oder S/390-Assembler geschrieben sind.

CharAttrLength
Typ: MQLONG - Eingabe

Dies ist die Länge des Parameters *CharAttrs* in Byte.

CharAttrLength muss mindestens der Summe der Längen der angeforderten Zeichenattribute entsprechen (siehe *Selectors*). Null ist ein gültiger Wert.

CharAttrs
Typ: MQCHAR × *CharAttrLength* - Ausgabe

Dies ist der Puffer, in dem die Zeichenattribute in verketteter Form zurückgegeben werden. Die Länge des Puffers wird mit dem Parameter *CharAttrLength* festgelegt.

Zeichenattribute werden in derselben Reihenfolge wie die MQCA_*-Selektoren im Parameter *Selectors* zurückgegeben. Die Länge der einzelnen Attributzeichenfolgen ist für jedes Attribut festgelegt (siehe *Selectors*) und der darin enthaltene Wert wird bei Bedarf nach rechts mit Leerzeichen aufgefüllt. Sie können einen Puffer bereitstellen, der größer ist als erforderlich, um alle angeforderten Zeichenattribute und Auffüllzeichen aufzunehmen. Die Byte jenseits des letzten zurückgegebenen Attributwerts bleiben unverändert.

Wenn *Hobj* eine Warteschlange angibt, aber der Attributselektor für den zugehörigen Warteschlangentyp ungültig ist, wird eine vollständig aus Sternen (*) bestehende Zeichenfolge zurückgegeben. Der Stern wird als der Wert dieses Attributs in *CharAttrs* zurückgegeben.

Wenn der Parameter *CharAttrLength* oder *SelectorCount* auf null gesetzt ist, wird *CharAttrs* nicht referenziert. In diesem Fall kann die Parameteradresse null sein, die von Programmen übergeben wird, die in C oder S/390-Assembler geschrieben sind.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine zu meldende Ursache.

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_CHAR_ATTRS_TOO_SHORT
(2008, X'7D8') Nicht genug Speicherplatz für Zeichenattribute zulässig.

MQRC_INT_ATTR_COUNT_TOO_SMALL
(2022, X'7E6') Nicht genug Speicherplatz für Ganzzahlenattribute zulässig.

MQRC_SELECTOR_NOT_FOR_TYPE
(2068, X'814') Selektor ungültig für Warteschlangentyp.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Länge von Zeichenattributen ist ungültig.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Zeichenfolge für Zeichenattribute ist ungültig.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung wurde von CICS abgelehnt.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig.

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Zähler von Ganzzahlenattributen ungültig.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Feldgruppe für Ganzzahlenattribute ungültig.

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') Warteschlange nicht für Abfrage geöffnet.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Zähler von Selektoren ungültig.

MQRC_SELECTOR_ERROR
(2067, X'813') Attributselektor ungültig.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Zähler von Selektoren zu groß.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes

Hinweise zur Verwendung

1. Die zurückgegebenen Werte sind eine Momentaufnahme der ausgewählten Attribute. Es gibt keine Garantie, dass die Attribute unverändert bleiben, bevor die Anwendung auf die zurückgegebenen Werte reagieren kann.
2. Beim Öffnen einer Modellwarteschlange wird eine dynamische lokale Warteschlange erstellt. Eine dynamische lokale Warteschlange wird auch dann erstellt, wenn Sie die Modellwarteschlange öffnen, um deren Attribute abzurufen.

   Die Attribute der dynamischen Warteschlange sind größtenteils dieselben wie die der Modellwarteschlange zum Zeitpunkt der Erstellung der dynamischen Warteschlange. Wenn Sie anschließend den Aufruf MQINQ auf diese Warteschlange anwenden, gibt der Warteschlangenmanager die Attribute der dynamischen Warteschlange und nicht die der Modellwarteschlange zurück. Im Abschnitt Tabelle 573 auf Seite 835 finden Sie Informationen darüber, welche Attribute der Modellwarteschlange von der dynamischen Warteschlange übernommen werden.
3. Wenn das abgefragte Objekt eine Aliaswarteschlange ist, werden vom MQINQ-Aufruf die Attributwerte der Aliaswarteschlange zurückgegeben. Das sind nicht die Attribute der Basiswarteschlange oder des Themas, in das der Aliasname aufgelöst wird.
4. Wenn das abgefragte Objekt eine Clusterwarteschlange ist, ist es davon abhängig, wie die Warteschlange geöffnet wird, welche Attribute abgefragt werden können:

   - Sie können eine Clusterwarteschlange für Abfragen und zusätzlich für eine oder mehrere Eingabe-, Anzeige- oder Einreihungsoperationen öffnen. Dazu muss es eine lokale Instanz der Clusterwarte-

schlange geben, damit die Warteschlange erfolgreich geöffnet werden kann. In diesem Fall können diejenigen Attribute abgefragt werden, die für lokale Warteschlangen gültig sind.

Wenn die Clusterwarteschlange ohne Angabe von Eingabe, Durchsuchen oder Festlegen geöffnet ist, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück, wenn Sie versuchen, Attribute abzufragen, die ausschließlich für lokale Warteschlangen und nicht für Clusterwarteschlangen gültig sind.

- Sie können eine Clusterwarteschlange zum Abfragen öffnen, während Sie den Basiswarteschlangenmanagernamen des verbundenen Warteschlangenmanagers übergeben.

  Dazu muss es eine lokale Instanz der Clusterwarteschlange geben, damit die Warteschlange erfolgreich geöffnet werden kann. Wenn der Basiswarteschlangenmanager nicht übergeben wird, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück, wenn Sie versuchen, Attribute abzufragen, die nur für lokale Warteschlangen und nicht für Clusterwarteschlangen gültig sind.
- Wird die Clusterwarteschlange nur für Abfragen oder für Abfragen und Ausgaben geöffnet, können nur die aufgelisteten Attribute abgefragt werden. Dass Attribut **QType** hat in diesem Fall den Wert MQQT_CLUSTER:
  - MQCA_Q_DESC
  - MQCA_Q_NAME
  - MQIA_DEF_BIND
  - MQIA_DEF_PERSISTENCE
  - MQIA_DEF_PRIORITY
  - MQIA_INHIBIT_PUT
  - MQIA_Q_TYPE

  Sie können die Clusterwarteschlange ohne festgelegte Bindung öffnen. Öffnen Sie sie mit dem Aufruf MQOPEN, für den MQOO_BIND_NOT_FIXED angegeben ist. Alternativ können Sie MQOO_BIND_AS_Q_DEF angeben und das Attribut **DefBind** der Warteschlange auf MQBND_BIND_NOT_FIXED setzen. Wenn Sie eine Clusterwarteschlange ohne feste Bindung öffnen, fragen aufeinanderfolgende MQINQ-Aufrufe möglicherweise verschiedene Instanzen der Clusterwarteschlange ab. Allerdings haben alle Instanzen normalerweise dieselben Attributwerte.
- Es kann ein Aliaswarteschlangenobjekt für einen Cluster definiert werden. Da TARGTYPE und TARGET keine Clusterattribute sind, kennt der Prozess, der einen MQOPEN-Prozess für die Aliaswarteschlange ausführt, nicht das Objekt, in das der Aliasname aufgelöst wird.

  Beim ersten MQOPEN-Prozess wird die Aliaswarteschlange in einen Warteschlangenmanager und eine Warteschlange im Cluster aufgelöst. Die Namensauflösung wird auf dem fernen Warteschlangenmanager wiederholt und erst dort wird der TARGTPYE der Aliaswarteschlange aufgelöst.

  Wenn die Aliaswarteschlange in ein Themenalias aufgelöst wird, erfolgt die Veröffentlichung von Nachrichten, die in die Aliaswarteschlange gestellt werden, auf diesem fernen Warteschlangenmanager.

Siehe Clusterwarteschlangen

5. Sie könnten eine Anzahl von Attributen abfragen und anschließend einige davon unter Verwendung des MQSET-Aufrufs setzen. Positionieren Sie die festzulegenden Attribute zur Programmierung der Abfrage und zum wirksamen Festlegen an den Anfang der Selektor-Arrays. Auf diese Weise können dieselben Arrays mit verringertem Zähler für MQSET verwendet werden.
6. Falls mehrere Warnungen zurückgegeben werden (siehe Parameter *CompCode*), ist der erste Ursachencode in der folgenden Liste der gültige Ursachencode:
   a. MQRC_SELECTOR_NOT_FOR_TYPE
   b. MQRC_INT_ATTR_COUNT_TOO_SMALL
   c. MQRC_CHAR_ATTRS_TOO_SHORT
7. Die folgenden Abschnitte enthalten Informationen zu Objektattributen:

- „Attribute für Warteschlangen” auf Seite 833
- „Attribute für Namenslisten” auf Seite 867
- „Attribute für Prozessdefinitionen” auf Seite 869
- „Attribute für den Warteschlangenmanager” auf Seite 797

C-Aufruf

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
     CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                        INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                        CHARATTRS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

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

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Abfragen von Nachrichteneigenschaften

Der Aufruf MQINQMP gibt den Wert einer Eigenschaft einer Nachricht zurück.

Syntax

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* muss mit der Verbindungskennung übereinstimmen, die verwendet wurde, um die im Parameter *Hmsg* angegebene Nachrichtenkennung zu erstellen.

Wenn die Nachrichtenkennung mit MQHC_UNASSOCIATED_HCONN erstellt wurde, muss eine gültige Verbindung in dem Thread erstellt werden, der eine Eigenschaft der Nachrichtenkennung abfragt. Andernfalls schlägt der Aufruf mit MQRC_CONNECTION_BROKEN fehl.

Hmsg
Typ: MQHMSG - Eingabe

Dies ist die abzufragende Nachrichtenkennung. Der Wert wurde von einem vorherigen **MQCRTMH**-Aufruf zurückgegeben.

InqPropOpts
Typ: MQIMPO - Ein-/Ausgabe

Details hierzu finden Sie unter dem MQIMPO-Datentyp.

Name
Typ: MQCHARV - Ein-/Ausgabe

Der Name der abzufragenden Eigenschaft.

Wenn keine Eigenschaft mit diesem Namen gefunden wird, schlägt der Aufruf mit dem Ursachencode MQRC_PROPERTY_NOT_AVAILABLE fehl.

Sie können das Prozentzeichen (%) als Platzhalterzeichen am Ende des Eigenschaftsnamens verwenden. Der Platzhalter steht für null oder mehr Zeichen, einschließlich des Punktes (.). Auf diese Weise kann die Anwendung eine große Anzahl von Eigenschaften abfragen. Rufen Sie MQINQMP mit der Option MQIMPO_INQ_FIRST auf, um die erste übereinstimmende Eigenschaft abzurufen, und rufen Sie MQINQMP dann erneut mit der Option MQIMPO_INQ_NEXT auf, um die nächste übereinstimmende Eigenschaft abzurufen. Wenn keine weiteren übereinstimmenden Eigenschaften verfügbar sind, schlägt der Aufruf mit MQRC_PROPERTY_NOT_AVAILABLE fehl. Wird das Feld *ReturnedName* der Struktur InqPropOpts mit einer Adresse oder einem Offset für den zurückgegebenen Namen der Eigenschaft initialisiert, wird dieser Schritt nach der Rückgabe von MQINQMP mit dem Namen der übereinstimmenden Eigenschaft ausgeführt. Wenn das Feld *VSBufSize* für *ReturnedName* in der InqPropOpts-Struktur kleiner ist als die Länge des zurückgegebenen Eigenschaftsnamens, wird der Beendigungscode auf MQCC_FAILED mit dem Ursachencode MQRC_PROPERTY_NAME_TOO_BIG gesetzt.

Eigenschaften, die Synonyme haben, werden wie folgt zurückgegeben:

1. Eigenschaften mit dem Präfix "mqps." werden als WebSphere MQ-Eigenschaftsname zurückgegeben. Beispielsweise ist "MQTopicString" der zurückgegebene Name, nicht "mqps.Top"
2. Eigenschaften mit dem Präfix "jms." oder "mcd." werden als JMS-Headerfeldname zurückgegeben. Beispielsweise ist "JMSExpiration" der zurückgegebene Name, nicht "jms.Exp".
3. Eigenschaften mit dem Präfix "usr." werden ohne dieses Präfix zurückgegeben. Beispielsweise wird "Color" zurückgegeben, nicht "usr.Color".

Eigenschaften mit Synonymen werden nur einmal zurückgegeben.

In der Programmiersprache C werden die folgenden Makrovariablen für die Einstellung aller Eigenschaften und dann für alle Eigenschaften definiert, die mit „usr.“ beginnen:

MQPROP_INQUIRE_ALL
Fragt alle Eigenschaften der Nachricht ab.

MQPROP_INQUIRE_ALL kann wie folgt verwendet werden:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR
Erkundigen Sie sich nach allen Eigenschaften der Nachricht, die mit "usr." starten. Der zurückgegebene Name wird ohne das Präfix "usr." zurückgegeben.

Wenn MQIMP_INQ_NEXT angegeben ist, sich der Name jedoch seit dem letzten Aufruf geändert hat oder dies der erste Aufruf ist, wird MQIMPO_INQ_FIRST eingeschlossen.

Weitere Informationen zur Verwendung von Eigenschaftsnamen finden Sie in den Abschnitten Eigenschaftsnamen und Einschränkungen bei Eigenschaftsnamen.

PropDesc
Typ: MQPD - Ausgabe

Über diese Struktur werden die Attribute einer Eigenschaft definiert, einschließlich, was geschieht, wenn die Eigenschaft nicht unterstützt wird, zu welchem Nachrichtenkontext die Nachricht gehört und in welche Nachrichten die Eigenschaft kopiert werden soll. Details zu dieser Struktur finden Sie im Abschnitt MQPD.

Type
Typ: MQLONG - Eingabe/Ausgabe

Bei Rückgabe des MQINQMP-Aufrufs wird dieser Parameter auf den Datentyp von *Value* gesetzt. Dabei kann es sich um folgende Datentypen handeln:

MQTYPE_BOOLEAN
Ein boolescher Wert.

MQTYPE_BYTE_STRING
Eine Bytefolge.

MQTYPE_INT8
Eine 8 Bit lange ganze Zahl mit Vorzeichen.

MQTYPE_INT16
Eine 16 Bit lange ganze Zahl mit Vorzeichen.

MQTYPE_INT32
Eine 32-Bit-Ganzzahl mit Vorzeichen.

MQTYPE_INT64
Eine 64-Bit-Ganzzahl mit Vorzeichen.

MQTYPE_FLOAT32
Eine 32 Bit lange Gleitkommazahl.

MQTYPE_FLOAT64
Eine 64 Bit lange Gleitkommazahl.

MQTYPE_STRING
Eine Zeichenfolge.

MQTYPE_NULL
Die Eigenschaft ist vorhanden, hat aber einen Nullwert.

Wenn der Datentyp des Eigenschaftswerts nicht erkannt wird, wird MQTYPE_STRING zurückgegeben und eine Zeichenfolgedarstellung des Werts wird im Bereich *Value* abgelegt. Eine Zeichenfolgedarstellung des Datentyps finden Sie im Feld *TypeString* des Parameters *InqPropOpts*. Ein Warnbeendigungscode wird mit der Ursache MQRC_PROP_TYPE_NOT_SUPPORTED zurückgegeben.

Darüber hinaus wird die Konvertierung des Eigenschaftswerts angefordert, wenn die Option MQIMPO_CONVERT_TYPE angegeben ist. Verwenden Sie *Type* als Eingabe, um den Datentyp anzugeben, als der die Eigenschaft zurückgegeben werden soll. Details zur Datentypkonvertierung finden Sie in der Beschreibung der Option MQIMPO_CONVERT_TYPE der MQIMPO-Struktur.

Wenn Sie die Typumwandlung nicht anfordern, können Sie bei der Eingabe den folgenden Wert verwenden:

MQTYPE_AS_SET
Der Wert der Eigenschaft wird zurückgegeben, ohne dass der Datentyp konvertiert wird.

ValueLength
Typ: MQLONG - Eingabe

Die Länge des Bereichs "Value" in Bytes. Geben Sie für Eigenschaften, für die der Wert nicht zurückgegeben werden muss, null an. Dies könnten Eigenschaften sein, die von einer Eigenschaft so konfiguriert sind, dass sie einen Nullwert oder eine leere Zeichenfolge haben. Geben Sie auch Null an, wenn die Option MQIMPO_QUERY_LENGTH angegeben wurde. In diesem Fall wird kein Wert zurückgegeben.

Wert
Typ: MQBYTEx*ValueLength* - Ausgabe

Dies ist der Bereich, in dem sich der abgefragte Eigenschaftswert befindet. Der Puffer muss auf einen auf den zurückzugebenden Wert abgestimmten Grenzwert ausgerichtet werden. Andernfalls kann es zu einem Fehler kommen, wenn später auf den Wert zugegriffen wird.

Wenn *ValueLength* kleiner ist als die Länge des Eigenschaftswerts, wird so viel wie möglich des Eigenschaftswerts in *Value* verschoben und der Aufruf schlägt mit dem Beendigungscode MQCC_FAILED und dem Ursachencode MQRC_PROPERTY_VALUE_TOO_BIG fehl.

Der Zeichensatz der Daten in *Value* wird durch das Feld ReturnedCCSID im Parameter InqPropOpts angegeben. Die Codierung der Daten in *Value* wird durch das Feld isReturnedEncoding im Parameter InqPropOpts angegeben.

In der Programmiersprache C ist der Parameter als pointer-to-void deklariert; als Parameter kann die Adresse eines jeden beliebigen Datentyps angegeben werden.

Wenn der Parameter *ValueLength* auf Null gesetzt ist, gibt es keinen Verweis auf *Value* und der von Programmen übergebene Wert, die in C oder System/390-Assembler geschrieben sind, kann Null sein.

DataLength
Typ: MQLONG - Ausgabe

Dies ist die Länge des Eigenschaftswerts in Byte, der im Bereich *Value* zurückgegeben wird.

Wenn *DataLength* kleiner ist als die Länge des Eigenschaftswerts, wird *DataLength* bei der Rückgabe vom Aufruf MQINQMP trotzdem ausgefüllt. Auf diese Weise kann die Anwendung die für die Aufnahme des Eigenschaftswerts erforderliche Puffergröße bestimmen und anschließend den Aufruf mit einem Puffer entsprechender Größe erneut ausgeben.

Die folgenden Werte können auch zurückgegeben werden.

Wenn der Parameter *Type* auf MQTYPE_STRING oder MQTYPE_BYTE_STRING gesetzt ist:

MQVL_EMPTY_STRING
Die Eigenschaft existiert, enthält aber keine Zeichen oder Byte.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_PROP_NAME_NOT_CONVERTED
(2492, X'09BC') Zurückgegebener Eigenschaftsname nicht konvertiert.

MQRC_PROP_VALUE_NOT_CONVERTED
(2466, X'09A2') Eigenschaftswert nicht konvertiert.

MQRC_PROP_TYPE_NOT_SUPPORTED
(2467, X'09A3') Eigenschaftsdatentyp wird nicht unterstützt.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975') Ein MQRFH2-Ordner, der Eigenschaften enthält, konnte nicht analysiert werden.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'086D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_ERROR
(2004, X'07D4') Wertparameter ungültig.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Längenparameter des Werts nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') Datenlängenparameter nicht gültig.

MQRC_IMPO_ERROR
(2464, X'09A0') Struktur zum Abfragen der Optionen für Nachrichteneigenschaften nicht gültig.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennung nicht gültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07F8') Optionen nicht gültig oder nicht konsistent.

MQRC_PD_ERROR
(2482, X'09B2') Struktur des Eigenschaftsdeskriptors nicht gültig.

MQRC_PROP_CONV_NOT_SUPPORTED
(2470, X'09A6') Konvertierung vom tatsächlichen in den angeforderten Datentyp nicht unterstützt.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Ungültiger Eigenschaftsname.

MQRC_PROPERTY_NAME_TOO_BIG
(2465, X'09A1') Eigenschaftsname zu groß für zurückgegebenen Namenspuffer.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7) Eigenschaft nicht verfügbar.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Eigenschaftswert zu groß für den Bereich "Value".

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Zahlenformatfehler in Wertdaten gefunden.

MQRC_PROPERTY_TYPE_ERROR
(2473, X'09A9') Ungültiger angeforderter Eigenschaftstyp.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') ID des codierten Zeichensatzes des Eigenschaftsnamens nicht gültig.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'0871') Nicht genug Speicher verfügbar.

MQRC_UNEXPECTED_ERROR
(2195, X'0893') Unerwarteter Fehler aufgetreten.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C-Aufruf

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDESC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQINQMP,(HCONN,HMSG,INQMSGOPTS,NAME,PROPDESC,TYPE,
VALUELENGTH,VALUE,DATALENGTH,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Konvertieren von Nachrichtenkennungen in Puffer

Der Aufruf MQMHBUF konvertiert eine Nachrichtenkennung in einen Puffer und ist die Umkehrfunktion des Aufrufs MQBUFMH.

Syntax

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* muss mit der Verbindungskennung übereinstimmen, die verwendet wurde, um die im Parameter *Hmsg* angegebene Nachrichtenkennung zu erstellen.

Wenn die Nachrichtenkennung mit MQHC_UNASSOCIATED_HCONN erstellt wurde, muss eine gültige Verbindung in dem Thread erstellt werden, der die Nachrichtenkennung löscht. Wird keine gültige Verbindung hergestellt, schlägt der Aufruf mit MQRC_CONNECTION_BROKEN fehl.

Hmsg

Typ: MQHMSG - Eingabe

Dies ist die Nachrichtenkennung, für die ein Puffer erforderlich ist. Der Wert wurde durch einen früheren MQCRTMH-Aufruf zurückgemeldet.

MsgHBufOpts

Typ: MQMHBO - Eingabe

Über die MQMHBO-Struktur können Anwendungen Optionen zur Steuerung der Erstellung von Puffern aus Nachrichtenkennungen festlegen.

Weitere Informationen finden Sie im Artikel „MQMHBO – Nachrichtenhandle-zu-Puffer-Optionen” auf Seite 460.

Name
Typ: MQCHARV - Eingabe

Der Name der Eigenschaft oder Eigenschaften, die im Puffer abgelegt werden sollen.

Wenn keine Eigenschaft gefunden wird, die mit diesem Namen übereinstimmt, schlägt der Aufruf mit MQRC_PROPERTY_NOT_AVAILABLE fehl.

Sie können ein Platzhalterzeichen verwenden, um mehr als eine Eigenschaft an den Puffer zu übergeben. Dazu verwenden Sie das Platzhalterzeichen '%' am Ende des Eigenschaftsnamens. Dieses Platzhalterzeichen stimmt mit null oder mehr Zeichen überein, einschließlich des .

In der Programmiersprache C sind die folgenden Makrovariablen definiert, um alle Eigenschaften und alle Eigenschaften, die mit 'usr' beginnen, abzufragen:

MQPROP_INQUIRE_ALL
Reiht alle Eigenschaften der Nachricht in den Puffer ein

MQPROP_INQUIRE_ALL_USR
Reiht alle Eigenschaften der Nachricht, die mit 'usr.' beginnen, in den Puffer ein.

Weitere Informationen zur Verwendung von Eigenschaftsnamen finden Sie in den Abschnitten Eigenschaftsnamen und Einschränkungen bei Eigenschaftsnamen.

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Die Struktur *MsgDesc* beschreibt den Inhalt des Pufferbereichs.

Bei der Ausgabe sind die Felder *Encoding*, *CodedCharSetId* und *Format* so definiert, dass sie die Codierung, die Zeichensatzkennung und das Format der Daten im Pufferbereich, wie vom Aufruf geschrieben, korrekt beschreiben.

Die Daten in dieser Struktur liegen im Zeichensatz und in der Codierung der Anwendung vor.

BufferLength
Typ: MQLONG - Eingabe

BufferLength ist die Länge des Pufferbereichs in Bytes.

Buffer
Typ: MQBYTExBufferLength - Ausgabe

Buffer definiert den Bereich, der die Nachrichteneigenschaften enthalten soll. Sie müssen den Puffer an einer 4-Byte-Grenze ausrichten.

Wenn *BufferLength* kleiner ist als die Länge, die zum Speichern der Eigenschaften in *Buffer* erforderlich ist, schlägt MQMHBUF mit MQRC_PROPERTY_VALUE_TOO_BIG fehl.

Der Inhalt des Puffers kann sich auch ändern, wenn der Aufruf fehlschlägt.

DataLength
Typ: MQLONG - Ausgabe

DataLength ist die Länge der zurückgegebenen Eigenschaften im Puffer in Bytes. Wenn der Wert null ist, haben keine Eigenschaften mit dem in *Name* angegebenen Wert übereingestimmt und der Aufruf schlägt mit dem Ursachencode MQRC_PROPERTY_NOT_AVAILABLE fehl.

Wenn *BufferLength* kleiner ist als die Länge, die zum Speichern der Eigenschaften im Puffer erforderlich ist, schlägt der Aufruf MQMHBUF mit MQRC_PROPERTY_VALUE_TOO_BIG fehl, es wird jedoch trotzdem ein Wert in *DataLength* eingegeben. Dadurch kann die Anwendung die Größe des Puffers ermitteln, der für die Eigenschaften erforderlich ist, und den Aufruf dann erneut mit der erforderlichen *BufferLength* ausgeben.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_MHBO_ERROR
(2501, X'095C') Struktur der Nachrichtenkennung-zu-Puffer-Optionen nicht gültig.

MQRC_BUFFER_ERROR
(2004, X'07D4') Pufferparameter nicht gültig.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Pufferlängenparameter nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Verbindung zum Warteschlangenmanager nicht mehr vorhanden.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') Datenlängenparameter nicht gültig.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennung nicht gültig.

MQRC_MD_ERROR
(2026, X'07EA') Nachrichtendeskriptor nicht gültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Eigenschafsname ist nicht gültig.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Eigenschaft nicht verfügbar.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') BufferLength-Wert für angegebene Eigenschaften zu klein.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C-Aufruf

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Hinweise zur Verwendung

MQMHBUF konvertiert eine Nachrichtenkennung in einen Puffer.

Sie können den Aufruf mit einem MQGET API-Exit verwenden, um mithilfe der Nachrichteneigenschaften-APIs auf bestimmte Eigenschaften zuzugreifen und diese dann in einem Puffer an eine Anwendung zurückgeben, die für die Verwendung von MQRFH2-Headern anstelle von Nachrichtenkennungen konzipiert wurde.

Dieser Aufruf ist die Umkehrfunktion des MQBUFMH-Aufrufs, mit dem Sie Nachrichteneigenschaften aus einem Puffer in eine Nachrichtenkennung übergeben können.

Aufruf in COBOL

```
     CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                          BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
              DataLength, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQMHBUF,(HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
                BUFFER,DATALENGTH,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 – Objekt öffnen

Mit dem MQOPEN-Aufruf wird der Zugriff auf ein Objekt eingerichtet.

Folgende Objekttypen sind gültig:

- Warteschlange (einschließlich Verteilerlisten)
- Namensliste
- Prozessdefinition
- Warteschlangenmanager
- Thema

Syntax

MQOPEN (*Hconn*, *ObjDesc*, *Options*, *Hobj*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

In z/OS für CICS -Anwendungen und in IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und der folgende Wert für *Hconn*angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

ObjDesc
Typ: MQOD - Ein-/Ausgabe

Dies ist eine Struktur, die das zu öffnende Objekt angibt (Details siehe „MQOD - Objektdeskriptor" auf Seite 463).

Wenn das Feld *ObjectName* im Parameter *ObjDesc* den Namen einer Modellwarteschlange enthält, wird eine dynamische lokale Warteschlangemit den Attributen der Modellwarteschlange erstellt. Dies geschieht unabhängig davon, welche Optionen Sie im Parameter *Options* angeben. Nachfolgende Operationen, die den vom MQOPEN-Aufruf zurückgegebenen Parameter *Hobj* verwenden, werden für die neue dynamische Warteschlange und nicht für die Modellwarteschlange ausgeführt. Dies gilt auch für die MQINQ- und MQSET-Aufrufe. Der Name der Modellwarteschlange im Parameter *ObjDesc* wird durch den Namen der erstellten dynamischen Warteschlange ersetzt. Der Typ der dynamischen Warteschlange wird durch den Wert des Attributs *DefinitionType* der Modellwarteschlange bestimmt (siehe „Attribute für Warteschlangen" auf Seite 833). Informationen zu den anwendbaren Optionen zum Schließen von dynamischen Warteschlangen finden Sie in der Beschreibung des MQCLOSE-Aufrufs.

Optionen
Typ: MQLONG - Eingabe

Sie müssen mindestens eine der folgenden Optionen angeben:

- MQOO_BROWSE
- MQOO_INPUT_* (nur eine dieser Optionen)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_* (nur eine dieser Optionen)

Die folgende Tabelle enthält Details zu diesen Optionen. Bei Bedarf können weitere Optionen angegeben werden. Wenn mehrere Optionen erforderlich sind, können die Werte

- gemeinsam hinzugefügt werden (dieselbe Konstante nicht mehr als einmal hinzufügen) oder
- Mithilfe der Operation 'bitweises ODER' kombinieren (wenn die Programmiersprache Bitoperationen unterstützt).

Auf ungültige Kombinationen wird hingewiesen; alle anderen Kombinationen sind gültig. Es sind nur Optionen zulässig, die auf den Typ des mit *ObjDesc* angegebenen Objekts anwendbar sind. Die folgende Tabelle zeigt gültige MQOPEN-Optionen für Warteschlangen und Themen.

Option	Alias [1]	Lokal und Modell	Fern	Cluster (nicht lokal)	Verteilerliste	Thema
MQOO_INPUT_AS_Q_DEF	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_INPUT_SHARED	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_INPUT_EXCLUSIVE	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_OUTPUT	Ja	Ja	Ja	Ja	Ja	Ja
MQOO_BROWSE	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_CO_OP	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_INQUIRE	Ja	Ja	[2]	Ja	Nein	Nein

Option	Alias [1]	Lokal und Modell	Fern	Cluster (nicht lokal)	Verteilerliste	Thema
MQOO_SET	Ja	Ja	[2]	Nein	Nein	Nein
MQOO_BIND_ON_OPEN[3]	Ja	Ja	Ja	Ja	Ja	Nein
MQOO_BIND_NOT_FIXED[3]	Ja	Ja	Ja	Ja	Ja	Nein
MQOO_BIND_ON_GROUP[3]	Ja	Ja	Ja	Ja	Ja	Nein
MQOO_BIND_AS_Q_DEF[3]	Ja	Ja	Ja	Ja	Ja	Nein
MQOO_SAVE_ALL_CONTEXT	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_PASS_IDENTITY_CONTEXT	Ja	Ja	Ja	Ja	Ja	[4]
MQOO_PASS_ALL_CONTEXT	Ja	Ja	Ja	Ja	Ja	Ja
MQOO_SET_IDENTITY_CONTEXT	Ja	Ja	Ja	Ja	Ja	[4]
MQOO_SET_ALL_CONTEXT	Ja	Ja	Ja	Ja	Ja	Ja
MQOO_NO_READ_AHEAD	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_READ_AHEAD	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_READ_AHEAD_AS_Q_DEF	Ja	Ja	Nein	Nein	Nein	Nein
MQOO_ALTERNATE_USER_AUTHORITY	Ja	Ja	Ja	Ja	Ja	Ja
MQOO_FAIL_IF_QUIESCING	Ja	Ja	Ja	Ja	Ja	Ja
MQOO_RESOLVE_LOCAL_Q	Ja	Ja	Ja	Ja	Nein	Nein
MQOO_RESOLVE_LOCAL_TOPIC	Nein	Nein	Nein	Nein	Nein	Ja
MQOO_NO_MULTICAST	Nein	Nein	Nein	Nein	Nein	Ja

Anmerkung:

1. Die Gültigkeit von Optionen für Aliasnamen hängt von der Gültigkeit der Option für die Warteschlange ab, in die der Aliasname aufgelöst wird.
2. Diese Option ist nur für die lokale Definition einer fernen Warteschlange gültig.
3. Diese Option kann für jeden Warteschlangentyp angegeben werden, wird aber ignoriert, wenn die Warteschlange keine Clusterwarteschlange ist. Das Warteschlangenattribut *DefBind* überschreibt die Basiswarteschlange aber auch dann, wenn sich die Aliaswarteschlange nicht in einem Cluster befindet.
4. Diese Attribute können mit einem Thema verwendet werden, betreffen aber nur den Kontext, der für die beibehaltene Nachricht festgelegt ist, nicht die Kontextfelder, die an Subskribenten gesendet werden.

Zugriffsoptionen: Die folgenden Optionen steuern den Typ der Operationen, die für das Objekt ausgeführt werden können:

MQOO_INPUT_AS_Q_DEF
Die Warteschlange wird geöffnet, um Nachrichten abzurufen; der Zugriff erfolgt unter Verwendung des für die Warteschlange gesetzten Standardwertes.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen geöffnet. Der Zugriff erfolgt entweder gemeinsam oder exklusiv, abhängig vom Wert des Warteschlangenattributs *DefInputOpenOption* (Details siehe „Attribute für Warteschlangen” auf Seite 833).

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind.

MQOO_INPUT_SHARED
Öffnet eine Warteschlange zum Abrufen von Nachrichten mit gemeinsamem Zugriff.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen geöffnet. Der Aufruf kann erfolgreich ausgeführt werden, wenn die Warteschlange zuvor von dieser oder einer anderen Anwendung mit MQOO_INPUT_SHARED geöffnet wurde, schlägt jedoch mit Ursachencode MQRC_OBJECT_IN_USE fehl, wenn die Warteschlange zuvor mit MQOO_INPUT_EXCLUSIVE geöffnet wurde.

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind.

MQOO_INPUT_EXCLUSIVE
Öffnet eine Warteschlange zum Abrufen von Nachrichten mit exklusivem Zugriff.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen geöffnet. Der Aufruf schlägt mit Ursachencode MQRC_OBJECT_IN_USE fehl, wenn die Warteschlange zuvor von dieser oder einer anderen Anwendung für eine beliebige Art der Eingabe (MQOO_INPUT_SHARED oder MQOO_INPUT_EXCLUSIVE) geöffnet wurde.

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind.

MQOO_OUTPUT

Öffnet eine Warteschlange zum Einreihen von Nachrichten bzw. ein Thema oder eine Themenzeichenfolge zum Veröffentlichen von Nachrichten.

Die Warteschlange oder das Thema wird für nachfolgende MQPUT-Aufrufe geöffnet.

Ein MQOPEN-Aufruf mit dieser Option kann auch dann erfolgreich ausgeführt werden, wenn das Warteschlangenattribut *InhibitPut* auf MQQA_PUT_INHIBITED gesetzt ist (obwohl nachfolgende MQPUT-Aufrufe fehlschlagen, solange das Attribut auf diesen Wert gesetzt ist).

Diese Option ist für alle Typen von Warteschlangen, einschließlich Verteilerlisten, und Themen gültig.

Für diese Optionen gelten folgende Hinweise:

- Es kann nur eine dieser Optionen angegeben werden.
- Ein MQOPEN-Aufruf mit einer dieser Optionen kann auch dann erfolgreich ausgeführt werden, wenn das Warteschlangenattribut *InhibitGet* auf MQQA_GET_INHIBITED gesetzt ist (obwohl nachfolgende MQGET-Aufrufe fehlschlagen, solange das Attribut auf diesen Wert gesetzt ist).
- Wenn die Warteschlange als nicht gemeinsam nutzbar definiert ist (d. h., das Warteschlangenattribut *Shareability* ist auf den Wert MQQA_NOT_SHAREABLE gesetzt), werden Versuche, die Warteschlange für gemeinsamen Zugriff zu öffnen, als Versuche, die Warteschlange für exklusiven Zugriff zu öffnen, behandelt.
- Wenn eine Aliaswarteschlange mit einer dieser Optionen geöffnet wird, richtet sich die Überprüfung auf exklusive Nutzung (oder darauf, ob eine andere Anwendung die Warteschlange exklusiv nutzt) auf die Basiswarteschlange, in die der Aliasname aufgelöst wird.
- Diese Optionen sind ungültig, wenn *ObjectQMgrName* einen WS-Manager-Aliasnamen angibt; dies gilt auch dann, wenn der Wert des Attributs *RemoteQMgrName* in der lokalen Definition einer fernen Warteschlange, der zur WS-Manager-Aliasnamensumsetzung verwendet wird, der Name des lokalen Warteschlangenmanagers ist.

MQOO_BROWSE
Die Warteschlange wird geöffnet, um Nachrichten anzuzeigen.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen mit einer der folgenden Optionen geöffnet:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Dies ist auch dann zulässig, wenn die Warteschlange gerade für exklusiven Zugriff (MQOO_INPUT_EXCLUSIVE) geöffnet ist. Ein MQOPEN-Aufruf mit der Option MQOO_BROWSE erzeugt einen Anzeigecursor und positioniert ihn logisch vor der ersten Nachricht in der Warteschlange (weitere Informationen siehe MQGMO - Options-Feld).

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind. Sie ist auch ungültig,

wenn *ObjectQMgrName* einen WS-Manager-Aliasnamen angibt; dies gilt auch dann, wenn der Wert des Attributs *RemoteQMgrName* in der lokalen Definition einer fernen Warteschlange, der zu WS-Manager-Aliasnamensumsetzung verwendet wird, der Name des lokalen Warteschlangenmanagers ist.

MQOO_CO_OP
Öffnet ein Objekt als kooperierendes Mitglied der Gruppe von Kennungen.

Diese Option ist nur zusammen mit der Option MQOO_BROWSE gültig. Wenn Sie ohne MQOO_BROWSE angegeben wird, gibt der MQOPEN den Fehler MQRC_OPTIONS_ERROR zurück.

Die zurückgegebene Kennung wird als Mitglied einer kooperierenden Gruppe von Kennungen für nachfolgende MQGET-Aufrufe mit einer der folgenden Optionen betrachtet:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind.

MQOO_INQUIRE
Öffnet ein Objekt zum Abfragen von Attributen.

Die Warteschlange, Namensliste, Prozessdefinition oder der Warteschlangenmanager wird zur Verwendung mit nachfolgenden MQINQ-Aufrufen geöffnet.

Diese Option ist für alle Objekttypen außer Verteilerlisten gültig. Sie ist ungültig, wenn *ObjectQMgrName* einen WS-Manager-Aliasnamen angibt; dies gilt auch dann, wenn der Wert des Attributs *RemoteQMgrName* in der lokalen Definition einer fernen Warteschlange, der zur WS-Manager-Aliasnamensumsetzung verwendet wird, der Name des lokalen Warteschlangenmanagers ist.

MQOO_SET
Die Warteschlange wird geöffnet, um Attribute zu setzen.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQSET-Aufrufen geöffnet.

Diese Option ist für alle Warteschlangentypen außer Verteilerlisten gültig. Sie ist ungültig, wenn *ObjectQMgrName* den Namen einer lokalen Definition einer fernen Warteschlange angibt; dies gilt auch dann, wenn der Wert des Attributs *RemoteQMgrName* in der lokalen Definition einer fernen Warteschlange, der zur WS-Manager-Aliasnamensumsetzung verwendet wird, der Name des lokalen Warteschlangenmanagers ist.

Bindeoptionen: Die folgenden Optionen sind gültig, wenn das zu öffnende Objekt eine Clusterwarteschlange ist. Diese Optionen steuern die Bindung der Warteschlangenkennung an eine Instanz der Clusterwarteschlange:

MQOO_BIND_ON_OPEN
Der lokale Warteschlangenmanager bindet die Warteschlangenkennung an eine Instanz der Zielwarteschlange, wenn die Warteschlange geöffnet wird. Dies hat zur Folge, dass alle Nachrichten, die mit dieser Kennung eingereiht werden, an dieselbe Instanz der Zielwarteschlange und über dieselbe Route gesendet werden.

Diese Option ist nur für Warteschlangen gültig und betrifft nur Clusterwarteschlangen. Die Option wird ignoriert, wenn sie für eine Warteschlange angegeben wird, die keine Clusterwarteschlange ist.

MQOO_BIND_NOT_FIXED
Diese Option beendet die Bindung der Warteschlangenkennung an die Instanz der Zielwarteschlange durch den lokalen Warteschlangenmanager. Dies hat zur Folge, dass nachfolgende MQPUT-Aufrufe, die diese Kennung verwenden, die Nachrichten an *verschiedene* Instanzen der Zielwarteschlange senden oder sie zwar an dieselbe Instanz senden, aber auf verschiedenen Routen. Es besteht außerdem die Möglichkeit, dass die ausgewählte Instanz später vom lokalen

Warteschlangenmanager, von einem fernen Warteschlangenmanager oder von einem Nachrichtenkanalagenten gemäß den Netzbedingungen geändert wird.

Anmerkung: Client- und Serveranwendungen, die eine *Folge* von Nachrichten austauschen müssen, um eine Transaktion zu beenden, dürfen MQOO_BIND_NOT_FIXED (bzw. MQOO_BIND_AS_Q_DEF, wenn *DefBind* auf den Wert MQBND_BIND_NOT_FIXED gesetzt ist) nicht verwenden, weil nachfolgende Nachrichten innerhalb der Folge möglicherweise an verschiedene Instanzen der Serveranwendung gesendet werden.

Wenn MQOO_BROWSE oder eine der MQOO_INPUT_*-Optionen für eine Clusterwarteschlange angegeben wird, ist der Warteschlangenmanager gezwungen, die lokale Instanz der Clusterwarteschlange auszuwählen. Dies bedeutet, dass die Bindung der Warteschlangenkennung auch dann festgelegt ist, wenn MQOO_BIND_NOT_FIXED angegeben wird.

Wenn MQOO_INQUIRE zusammen mit MQOO_BIND_NOT_FIXED angegeben wird, fragen nachfolgende MQINQ-Aufrufe, die diese Kennung verwenden, möglicherweise verschiedene Instanzen der Clusterwarteschlange ab, obwohl normalerweise alle Instanzen dieselben Attributwerte haben.

MQOO_BIND_NOT_FIXED ist nur für Warteschlangen gültig und betrifft nur Clusterwarteschlangen. Die Option wird ignoriert, wenn sie für eine Warteschlange angegeben wird, die keine Clusterwarteschlange ist.

MQOO_BIND_ON_GROUP
Mit dieser Option kann eine Anwendung fordern, dass alle Nachrichten einer Nachrichtengruppe an dieselbe Zielinstanz übergeben werden.

Diese Option ist nur für Warteschlangen gültig und betrifft nur Clusterwarteschlangen. Die Option wird ignoriert, wenn sie für eine Warteschlange angegeben wird, die keine Clusterwarteschlange ist.

MQOO_BIND_AS_Q_DEF
Der lokale Warteschlangenmanager führt die Bindung der Warteschlangenkennung so durch, wie es durch das Warteschlangenattribut *DefBind* festgelegt ist. Der Wert dieses Attributs ist entweder MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED oder MQBND_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF ist der Standardwert, wenn MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED oder MQOO_BIND_ON_GROUP nicht angegeben sind.

MQOO_BIND_AS_Q_DEF unterstützt die Programmdokumentation. Diese Option ist nicht dazu gedacht, mit einer der beiden anderen Bindeoptionen verwendet zu werden, aber da sie den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

Kontextoptionen: Die folgenden Optionen steuern die Verarbeitung des Nachrichtenkontextes:

MQOO_SAVE_ALL_CONTEXT
Dieser Warteschlangenkennung werden Kontextinformationen zugeordnet. Die Informationen werden aus dem Kontext jeder Nachricht, die mit dieser Kennung abgerufen wird, zusammengestellt. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten Nachrichtenkontext und Steuerung von Kontextinformationen.

Die Kontextinformationen können an eine Nachricht übergeben werden, die anschließend mit MQPUT- oder MQPUT1-Aufrufen in eine Warteschlange gestellt werden. Weitere Informationen finden Sie in der Beschreibung der Optionen MQPMO_PASS_IDENTITY_CONTEXT und MQPMO_PASS_ALL_CONTEXT im Abschnitt „MQPMO - Nachrichteneinreihungsoptionen” auf Seite 484.

Solange eine Nachricht nicht erfolgreich abgerufen wurde, kann kein Kontext an eine Nachricht übergeben werden, die in eine Warteschlange gestellt wird.

Für eine Nachricht, die mit einer der MQGMO_BROWSE_*-Anzeigeoptionen abgerufen wird, werden keine Kontextinformationen gespeichert (obwohl die Kontextfelder im Parameter *MsgDesc* nach einer Anzeige gesetzt sind).

Diese Option ist nur für lokale, Alias- und Modellwarteschlangen gültig. Sie ist nicht gültig für ferne Warteschlangen, Verteilerlisten und Objekte, die keine Warteschlangen sind. Es muss eine der MQOO_INPUT_*-Optionen angegeben werden.

MQOO_PASS_IDENTITY_CONTEXT
Diese Option ermöglicht die Angabe der Option MQPMO_PASS_IDENTITY_CONTEXT im Parameter *PutMsgOpts*, wenn eine Nachricht in eine Warteschlange gestellt wird. Auf diese Weise erhält die Nachricht die Identitätskontextinformationen aus einer Eingabewarteschlange, die mit der Option MQOO_SAVE_ALL_CONTEXT geöffnet wurde. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten Nachrichtenkontext und Steuerung von Kontextinformationen.

Es muss die Option MQOO_OUTPUT angegeben werden.

Diese Option ist für alle Warteschlangentypen, einschließlich Verteilerlisten, gültig.

MQOO_PASS_ALL_CONTEXT
Diese Option ermöglicht die Angabe der Option MQPMO_PASS_ALL_CONTEXT im Parameter *PutMsgOpts*, wenn eine Nachricht in eine Warteschlange gestellt wird. Auf diese Weise erhält die Nachricht die Identitäts- und Ursprungskontextinformationen aus einer Eingabewarteschlange, die mit der Option MQOO_SAVE_ALL_CONTEXT geöffnet wurde. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten Nachrichtenkontext und Steuerung von Kontextinformationen.

Diese Option schließt die Option MQOO_PASS_IDENTITY_CONTEXT ein, die deshalb nicht angegeben werden muss. Es muss die Option MQOO_OUTPUT angegeben werden.

Diese Option ist für alle Warteschlangentypen, einschließlich Verteilerlisten, gültig.

MQOO_SET_IDENTITY_CONTEXT
Diese Option ermöglicht die Angabe der Option MQPMO_SET_IDENTITY_CONTEXT im Parameter *PutMsgOpts*, wenn eine Nachricht in eine Warteschlange gestellt wird. Auf diese Weise erhält die Nachricht die Identitätskontextinformationen, die im Parameter *MsgDesc*, der im MQPUT- oder MQPUT1-Aufruf angegeben ist, enthalten sind. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten Nachrichtenkontext und Steuerung von Kontextinformationen.

Diese Option schließt die Option MQOO_PASS_IDENTITY_CONTEXT ein, die deshalb nicht angegeben werden muss. Es muss die Option MQOO_OUTPUT angegeben werden.

Diese Option ist für alle Warteschlangentypen, einschließlich Verteilerlisten, gültig.

MQOO_SET_ALL_CONTEXT
Diese Option ermöglicht die Angabe der Option MQPMO_SET_ALL_CONTEXT im Parameter *PutMsgOpts*, wenn eine Nachricht in eine Warteschlange gestellt wird. Auf diese Weise erhält die Nachricht die Identitäts- und Ursprungskontextinformationen, die im Parameter *MsgDesc*, der im MQPUT- oder MQPUT1-Aufruf angegeben ist, enthalten sind. Weitere Informationen zum Nachrichtenkontext finden Sie in den Abschnitten Nachrichtenkontext und Steuerung von Kontextinformationen.

Diese Option schließt folgende Optionen ein, die deshalb nicht angegeben werden müssen:

- MQOO_PASS_IDENTITY_CONTEXT
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT

Es muss die Option MQOO_OUTPUT angegeben werden.

Diese Option ist für alle Warteschlangentypen, einschließlich Verteilerlisten, gültig.

Vorausleseoptionen:

Wenn Sie MQOPEN mit MQOO_READ_AHEAD aufrufen, aktiviert der WebSphere MQ-Client Vorauslesen nur unter bestimmten Bedingungen. Zu diesen Bedingungen gehören:

- Client und ferner Warteschlangenmanager müssen beide die WebSphere MQ Version 7 oder höher haben.

- Die Clientanwendung muss kompiliert und mit den WebSphere MQ-Clientbibliotheken mit Thread verknüpft sein.
- Der Clientkanal muss das TCP/IP-Protokoll verwenden.
- In den Client- und Serverkanaldefinitionen muss für den Kanal ein Wert ungleich null für den Parameter SHARECNV (gemeinsame Dialognutzung) angegeben sein.

Die folgenden Optionen steuern, ob nicht persistente Nachrichten an den Client gesendet werden, bevor eine Anwendung sie anfordert. Für Vorausleseoptionen gelten folgende Hinweise:

- Es kann nur eine dieser Optionen angegeben werden.
- Die Optionen sind nur für lokale, Alias- und Modellwarteschlangen gültig. Sie sind nicht gültig für ferne Warteschlangen, Verteilerlisten, Themen oder Warteschlangenmanager.
- Die Optionen sind nur anwendbar, wenn auch eine der Optionen MQOO_BROWSE, MQOO_INPUT_SHARED und MQOO_INPUT_EXCLUSIVE angegeben ist, wobei es jedoch kein Fehler ist, diese Optionen zusammen mit MQOO_INQUIRE oder MQOO_SET anzugeben.
- Wird die Anwendung nicht als IBM WebSphere MQ -Client ausgeführt, werden diese Optionen ignoriert.

MQOO_NO_READ_AHEAD
Nicht persistente Nachrichten werden nicht an den Client gesendet, bevor sie von einer Anwendung angefordert werden.

MQOO_READ_AHEAD
Nicht persistente Nachrichten werden an den Client gesendet, bevor eine Anwendung sie anfordert.

MQOO_READ_AHEAD_AS_Q_DEF
Das Vorausleseverhalten wird durch das Standardattribut für Vorauslesen, das für die zu öffnende Warteschlange festgelegt ist, bestimmt. Dies ist der Standardwert.

Weitere Optionen: Die folgenden Optionen steuern die Berechtigungsprüfung, die Versetzung des Warteschlangenmanagers in den Quiescemodus, das Auflösen des Namens der lokalen Warteschlange und Multicasting:

MQOO_ALTERNATE_USER_AUTHORITY
Das Feld *AlternateUserId* im Parameter *ObjDesc* enthält eine Benutzer-ID zur Überprüfung der Berechtigung des MQOPEN-Aufrufs. Der Aufruf kann nur erfolgreich ausgeführt werden, wenn die ID in *AlternateUserId* berechtigt ist, das Objekt mit den angegebenen Zugriffsoptionen zu öffnen, und zwar unabhängig davon, ob die Benutzer-ID, unter der die Anwendung ausgeführt wird, dazu berechtigt ist. Dies gilt jedoch nicht für angegebene Kontextoptionen, die immer anhand der Benutzer-ID überprüft werden, unter der die Anwendung ausgeführt wird.

Diese Option ist für alle Objekttypen gültig.

MQOO_FAIL_IF_QUIESCING
Der MQOPEN-Aufruf schlägt fehl, wenn sich der Warteschlangenmanager im Quiescemodus befindet.

Unter z/OSerzwingt diese Option für eine CICS -oder IMS -Anwendung auch das Fehlschlagen des MQOPEN-Aufrufs, wenn sich die Verbindung im Quiescemodus befindet.

Diese Option ist für alle Objekttypen gültig.

Informationen zu Clientkanälen finden Sie unter Übersicht über IBM WebSphere MQ MQI-Clients .

MQOO_RESOLVE_LOCAL_Q
Füllt das Feld ResolvedQName in der MQOD-Struktur mit dem Namen der lokalen Warteschlange, die geöffnet wurde. Entsprechend wird das Feld ResolvedQMgrName mit dem Namen des lokalen Warteschlangenmanagers gefüllt, der die lokale Warteschlange verwaltet. Wenn es sich um eine MQOD-Struktur kleiner als Version 3 handelt, wird MQOO_RESOLVE_LOCAL_Q ignoriert und kein Fehler zurückgegeben.

Die lokale Warteschlange wird immer zurückgegeben, wenn entweder eine lokale, Alias- oder Modellwarteschlange geöffnet wird. Dies ist jedoch nicht der Fall, wenn beispielsweise eine ferne Warteschlange oder eine nicht lokale Clusterwarteschlange ohne Angabe der Option MQOO_RESOLVE_LOCAL_Q geöffnet wird. Die Felder ResolvedQName und ResolvedQMgrName werden mit den Werten aus den Feldern RemoteQName und RemoteQMgrName, die in der Definition der fernen Warteschlange bzw. der ausgewählten fernen Clusterwarteschlange gefunden werden, gefüllt.

Wenn Sie MQOO_RESOLVE_LOCAL_Q beispielsweise beim Öffnen einer fernen Warteschlange angeben, enthält ResolvedQName die Übertragungswarteschlange, in die Nachrichten eingereiht werden. Das Feld ResolvedQMgrName wird mit dem Namen des lokalen Warteschlangenmanagers gefüllt, der die Übertragungswarteschlange verwaltet.

Wenn Sie zur Anzeige, Eingabe oder Ausgabe für eine Warteschlange berechtigt sind, besitzen Sie die erforderliche Berechtigung, dieses Attribut im MQOPEN-Aufruf anzugeben. Eine Sonderberechtigung ist nicht erforderlich.

Diese Option ist nur für Warteschlangen und Warteschlangenmanager gültig.

MQOO_RESOLVE_LOCAL_TOPIC
Füllt das Feld ResolvedQName in der MQOD-Struktur mit dem Namen des administrativen Themas, das geöffnet wurde.

MQOO_NO_MULTICAST
Veröffentlichungsnachrichten werden nicht mit Multicasting gesendet.

Diese Option ist nur zusammen mit der Option MQOO_OUTPUT gültig. Wenn sie ohne MQOO_OUTPUT angegeben wird, gibt MQOPEN den Fehler MQRC_OPTIONS_ERROR zurück.

Diese Option ist nur für ein Thema gültig.

Hobj
Typ: MQHOBJ - Ausgabe

Diese Kennung steht für den Zugriff, der für das Objekt eingerichtet wurde. Sie muss in nachfolgenden IBM WebSphere MQ-Aufrufen, die Operationen für das Objekt durchführen, angegeben werden. Die Kennung wird ungültig, wenn der MQCLOSE-Aufruf ausgegeben oder die Verarbeitungseinheit, die den Geltungsbereich der Kennung definiert, beendet wird.

Der Geltungsbereich der zurückgegebenen Objektkennung entspricht dem Geltungsbereich der Verbindungskennung, die im Aufruf angegeben ist. Informationen zum Geltungsbereich der Kennung finden Sie im Abschnitt MQCONN - Hconn-Parameter.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Aliasbasiswarteschlange kein gültiger Typ.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_NOT_AVAILABLE
(2345, X'929') Coupling-Facility nicht verfügbar

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') Berechtigungsprüfung für Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_ERROR
(2349, X'92D') Coupling-Facility-Struktur ungültig

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Listenüberschrift für Coupling-Facility-Struktur im Gebrauch

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Exit für Clusterauslastung fehlgeschlagen.

MQRC_CLUSTER_PUT_INHIBITED
(2268, X'8DC') PUT-Aufrufe für alle Warteschlangen im Cluster unterdrückt

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Clusternamensauflösung fehlgeschlagen.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Clusterressourcenfehler.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') Db2-Subsystem nicht verfügbar

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896') Standardübertragungswarteschlange nicht lokal.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897') Fehler bei Verwendung der Standardübertragungswarteschlange.

MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') Name der dynamischen Warteschlange ungültig

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') Keine weiteren Kennungen verfügbar.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

MQRC_NAME_IN_USE
(2201, X'899') Name im Gebrauch

MQRC_NAME_NOT_VALID_FOR_TYPE
(2194, X'892') Objektname für Objekttyp ungültig

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834') Objekt vorhanden

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OBJECT_IN_USE
(2042, X'7FA') Objekt bereits mit unzulässiger Kombination von Optionen geöffnet.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X'938') Objektebene nicht kompatibel

MQRC_OBJECT_NAME_ERROR
(2152, X'868') Objektname ungültig

MQRC_OBJECT_NOT_UNIQUE
(2343, X'927') Objekt nicht eindeutig

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869') Warteschlangenmanagername für Objekt ungültig

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Objektdatensätze ungültig.

MQRC_OBJECT_STRING_ERROR
(2441, X'0989') Objektzeichenfolgefeld ungültig

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Objekttyp ungültig.

MQRC_OD_ERROR
(2044, X'7FC') Objektdeskriptorstruktur ungültig.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Option für Objekttyp ungültig.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_PAGESET_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_Q_TYPE_ERROR
(2057, X'809') Warteschlangentyp ungültig.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Anzahl vorhandener Datensätze ungültig.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888') Name der fernen Warteschlange ungültig.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Antwortdatensätze ungültig.

MQRC_SECURITY_ERROR
(2063, X'80F') Es ist ein Sicherheitsfehler aufgetreten.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Es wurde ein MQOPEN-, MQPUT1- oder MQSUB-Aufruf ausgegeben, aber eine Auswahlzeichenfolge mit einem Syntaxfehler angegeben

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Aufruf vom Exit für Clusterauslastung zurückgewiesen.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822') Unbekannte Aliasbasiswarteschlange.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895') Unbekannte Standardübertragungswarteschlange.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unbekannter Objektname.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826') Unbekannter Objektwarteschlangenmanager.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827') Unbekannter ferner Warteschlangenmanager.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894') Unbekannte Übertragungswarteschlange.

MQRC_WRONG_CF_LEVEL
(2366, X'93E') Coupling-Facility-Struktur mit falscher Version.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Übertragungswarteschlange nicht lokal.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Übertragungswarteschlange mit falscher Verwendung.

Detaillierte Informationen zu diesen Codes finden Sie in folgenden Abschnitten:

- Ursachencodes für alle anderen IBM WebSphere MQ -Plattformen außer z/OS.

Allgemeine Hinweise zur Verwendung

1. Es wird eines der folgenden Objekte geöffnet:
   - Eine Warteschlange zum:
     - Abrufen oder Anzeigen von Nachrichten (mit dem MQGET-Aufruf)
     - Einreihen von Nachrichten (mit dem MQPUT-Aufruf)
     - Abfragen der Attribute der Warteschlange (mit dem MQINQ-Aufruf)
     - Festlegen der Attribute der Warteschlange (mit dem MQSET-Aufruf)

     Wenn die benannte Warteschlange eine Modellwarteschlange ist, wird eine dynamische lokale Warteschlange erstellt. Weitere Informationen finden Sie in der Beschreibung des Parameters *ObjDesc* im Abschnitt „MQOPEN – Objekt öffnen” auf Seite 728.

     Eine Verteilerliste ist ein besonderer Warteschlangenobjekttyp, der eine Liste mit Warteschlangen enthält. Sie kann geöffnet werden, um Nachrichten einzureihen, aber nicht, um Nachrichten abzurufen oder anzuzeigen oder um Attribute abzufragen oder festzulegen. Weitere Informationen siehe Hinweis 8.

     Eine Warteschlange mit `QSGDISP(GROUP)` ist ein besonderer Typ von Warteschlangendefinition, der nicht mit den MQOPEN- und MQPUT1-Aufrufen verwendet werden kann.
   - Eine Namensliste zum Abfragen der Namen der Warteschlangen in der Liste (mit dem MQINQ-Aufruf).
   - Eine Prozessdefinition zum Abfragen der Prozessattribute (mit dem MQINQ-Aufruf).
   - Der Warteschlangenmanager zum Abfragen der Attribute des lokalen Warteschlangenmanagers (mit dem MQINQ-Aufruf).
   - Ein Thema zum Veröffentlichen einer Nachricht (mit dem Aufruf MQPUT).
2. Eine Anwendung kann dasselbe Objekt mehrfach öffnen. Bei jedem Öffnen wird eine andere Objektkennung zurückgegeben. Jede Kennung, die zurückgegeben wird, kann für die Funktionen verwendet werden, für die der entsprechende Aufruf zum Öffnen ausgeführt wurde.
3. Wenn das zu öffnende Objekt keine Clusterwarteschlange ist, erfolgen alle Namensauflösungen innerhalb des lokalen Warteschlangenmanagers zum Zeitpunkt des MQOPEN-Aufrufs. Dies kann Folgendes einschließen:
   - Auflösung des Namens einer lokalen Definition einer fernen Warteschlange in den Namen des fernen Warteschlangenmanagers und in den Namen, unter dem die Warteschlange dem fernen Warteschlangenmanager bekannt ist
   - Auflösung des Namens des fernen Warteschlangenmanagers in den Namen einer lokalen Übertragungswarteschlange
   - Nur z/OS: Auflösung des Namens des fernen Warteschlangenmanagers in den Namen der gemeinsam genutzten Übertragungswarteschlange, die vom IGQ-Agenten verwendet wird (gilt nur, wenn die lokalen und fernen Warteschlangenmanager derselben Gruppe mit gemeinsamer Warteschlange angehören)
   - Aliasnamensauflösung in den Namen einer Basiswarteschlange oder eines Themenobjekts.

   Es ist jedoch zu beachten, dass sich nachfolgende MQINQ- oder MQSET-Aufrufe für die Kennung allein auf den Namen beziehen, der geöffnet wurde, und nicht auf das Objekt, das sich aus der erfolgten Namensauflösung ergibt. Wenn das geöffnete Objekt beispielsweise ein Alias ist, handelt es sich bei

den vom MQINQ-Aufruf zurückgegebenen Attributen um die des Alias und nicht um die Attribute der Basiswarteschlange, in die das Alias aufgelöst wird, oder eines Themenobjekts, in das das Alias aufgelöst wird.

Wenn das zu öffnende Objekt eine Clusterwarteschlange ist, kann die Namensauflösung zum Zeitpunkt des MQOPEN-Aufrufs erfolgen oder auf einen späteren Zeitpunkt verschoben werden. Der Punkt, an dem die Auflösung erfolgt, wird durch die MQOO_BIND_*-Optionen gesteuert, die im MQOPEN-Aufruf angegeben werden:

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED
- MQOO_BIND_AS_Q_DEF
- MQOO_BIND_ON_GROUP

Weitere Informationen zur Namensauflösung für Clusterwarteschlangen finden Sie im Abschnitt Namensauflösung.

4. Ein MQOPEN-Aufruf mit der Option MQOO_BROWSE erzeugt einen Anzeigecursor für die Verwendung mit MQGET-Aufrufen, in denen die Objektkennung und eine der Suchoptionen angegeben werden. Auf diese Weise kann die Warteschlange durchsucht werden, ohne ihren Inhalt zu ändern. Eine Nachricht, die bei der Suche gefunden wurde, kann mit der Option MQGMO_MSG_UNDER_CURSOR aus der Warteschlange entfernt werden.

   Indem mehrere MQOPEN-Anforderungen für dieselbe Warteschlange ausgegeben werden, können mehrere Anzeigecursor für eine einzelne Anwendung aktiv sein.
5. An Anwendungen, die von einem Auslösemonitor gestartet werden, wird der Name der Warteschlange übergeben, die der Anwendung bei ihrem Start zugeordnet wird. Dieser Warteschlangenname kann im Parameter *ObjDesc* zum Öffnen der Warteschlange angegeben werden. Weitere Informationen finden Sie im Abschnitt „MQTMC2 - Auslösenachricht 2 (Zeichenformat)" auf Seite 596.
6. Unter IBM i werden Anwendungen, die im Kompatibilitätsmodus aktiv sind, automatisch beim ersten durch die Anwendung ausgegebenen MQOPEN-Aufruf mit dem Warteschlangenmanager verbunden (falls die Anwendung nicht bereits über den MQCONN-Aufruf mit dem Warteschlangenmanager verbunden wurde).

   Anwendungen, die nicht im Kompatibilitätsmodus aktiv sind, müssen den MQCONN- oder MQCONNX-Aufruf ausgeben, um explizit eine Verbindung mit dem Warteschlangenmanager herzustellen, bevor ein Objekt mit dem MQOPEN-Aufruf geöffnet wird.

Vorausleseoptionen

Wenn Sie MQOPEN mit MQOO_READ_AHEAD aufrufen, aktiviert der WebSphere MQ-Client Vorauslesen nur unter bestimmten Bedingungen. Zu diesen Bedingungen gehören:

- Client und ferner Warteschlangenmanager müssen beide die WebSphere MQ Version 7 oder höher haben.
- Die Clientanwendung muss kompiliert und mit den WebSphere MQ-Clientbibliotheken mit Thread verknüpft sein.
- Der Clientkanal muss das TCP/IP-Protokoll verwenden.
- In den Client- und Serverkanaldefinitionen muss für den Kanal ein Wert ungleich null für den Parameter SHARECNV (gemeinsame Dialognutzung) angegeben sein.

Die folgenden Hinweise gelten für die Verwendung von Vorausleseoptionen.

1. Die Vorausleseoptionen sind nur anwendbar, wenn auch eine (und nur eine einzige) der Optionen MQOO_BROWSE, MQOO_INPUT_SHARED und MQOO_INPUT_EXCLUSIVE angegeben wird. Es wird kein Fehler ausgegeben, wenn Vorausleseoptionen zusammen mit der Option MQOO_ INQUIRE oder MQOO_SET angegeben werden.
2. Das Vorauslesen wird nicht wie angefordert aktiviert, wenn die im ersten MQGET-Aufruf verwendeten Optionen nicht für das Vorauslesen unterstützt werden. Außerdem wird das Vorauslesen inaktiviert,

wenn der Client eine Verbindung mit einem Warteschlangenmanager herstellt, der das Vorauslesen nicht unterstützt.

3. Wenn die Anwendung nicht als IBM WebSphere MQ -Client ausgeführt wird, werden Vorausleseoptionen ignoriert.

Clusterwarteschlangen

Die folgenden Hinweise gelten für die Verwendung von Clusterwarteschlangen.

1. Wenn eine Clusterwarteschlange zum ersten Mal geöffnet wird und der lokale Warteschlangenmanager nicht über ein vollständiges Warteschlangenmanager-Repository verfügt, ruft er Informationen zu der Clusterwarteschlange aus einem vollständigen Warteschlangenmanager-Repository ab. Wenn das Netz ausgelastet ist, kann es mehrere Sekunden dauern, bis der lokale Warteschlangenmanager die benötigten Informationen aus dem Warteschlangenmanager-Repository empfängt. Dies führt dazu, dass die Anwendung, die den MQOPEN-Aufruf ausgibt, möglicherweise bis zu 10 Sekunden warten muss, bis der MQOPEN-Aufruf die Steuerung zurückgibt. Wenn der lokale Warteschlangenmanager die benötigten Informationen zu der Clusterwarteschlange nicht innerhalb dieser Zeit empfängt, schlägt der Aufruf mit Ursachencode MQRC_CLUSTER_RESOLUTION_ERROR fehl.
2. Wenn eine Clusterwarteschlange geöffnet wird und sich im Cluster mehrere Instanzen der Warteschlange befinden, hängt es von den im MQOPEN-Aufruf angegebenen Optionen ab, welche Instanz geöffnet wird:
   - Wenn unter den angegebenen Optionen mindestens eine der folgenden Optionen ist:
     - MQOO_BROWSE
     - MQOO_INPUT_AS_Q_DEF
     - MQOO_INPUT_EXCLUSIVE
     - MQOO_INPUT_SHARED
     - MQOO_SET

     In diesem Fall muss es sich bei der geöffneten Instanz der Clusterwarteschlange um die lokale Instanz handeln. Falls keine lokale Instanz der Warteschlange vorhanden ist, schlägt der MQOPEN-Aufruf fehl.
   - Unter den angegebenen Optionen ist zwar keine der oben beschriebenen Optionen, aber eine oder beide der folgenden Optionen:
     - MQOO_INQUIRE
     - MQOO_OUTPUT

     In diesem Fall handelt es sich bei der geöffneten Instanz um die lokale Instanz, sofern eine vorhanden ist, und andernfalls um eine ferne Instanz (bei Verwendung der CLWLUSEQ-Standardwerte). Die vom Warteschlangenmanager ausgewählte Instanz kann jedoch von einem Exit für Clusterauslastung (falls vorhanden) geändert werden.
3. Wenn es eine Subskription für die Warteschlange gibt, diese aber nicht von einem vollständigen Repository bestätigt wird, ist das Objekt nicht im Cluster enthalten und der Aufruf schlägt mit Ursachencode MQRC_OBJECT_NAME fehl.

Weitere Informationen zu Clusterwarteschlangen finden Sie im Abschnitt Clusterwarteschlangen.

Verteilerlisten

Die folgenden Hinweise gelten für die Verwendung von Verteilerlisten.

Verteilerlisten werden in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie IBM WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

1. Felder in der MQOD-Struktur müssen beim Öffnen einer Verteilerliste auf folgende Werte gesetzt sein:
   - *Version* muss MQOD_VERSION_2 oder höher sein.

- *ObjectType* muss MQOT_Q sein.
- *ObjectName* muss ein Leerzeichen oder die Nullzeichenfolge sein.
- *ObjectQMgrName* muss ein Leerzeichen oder die Nullzeichenfolge sein.
- *RecsPresent* muss größer als null sein.
- Eines der Felder *ObjectRecOffset* und *ObjectRecPtr* muss gleich null und das andere ungleich null sein.
- Nur eines der Felder *ResponseRecOffset* und *ResponseRecPtr* kann ungleich null sein.
- Es müssen *RecsPresent* Objektdatensätze vorhanden sein, die entweder durch *ObjectRecOffset* oder *ObjectRecPtr* adressiert werden. Die Objektdatensätze müssen auf die Namen der Zielwarteschlangen gesetzt sein, die geöffnet werden sollen.
- Wenn eines der Felder *ResponseRecOffset* und *ResponseRecPtr* ungleich null ist, müssen *RecsPresent* Antwortdatensätze vorhanden sein. Sie werden vom Warteschlangenmanager gesetzt, wenn der Aufruf mit Ursachencode MQRC_MULTIPLE_REASONS beendet wird.

Es kann auch ein MQOD der Version 2 zum Öffnen einer einzelnen Warteschlange, die nicht in einer Verteilerliste enthalten ist, verwendet werden, indem sichergestellt wird, dass *RecsPresent* gleich null ist.

2. Im Parameter *Options* sind nur folgende Optionen zum Öffnen gültig:
   - MQOO_OUTPUT
   - MQOO_PASS_*_CONTEXT
   - MQOO_SET_*_CONTEXT
   - MQOO_ALTERNATE_USER_AUTHORITY
   - MQOO_FAIL_IF_QUIESCING
3. Bei den Zielwarteschlangen in der Verteilerliste kann es sich um lokale, Alias- oder ferne Warteschlangen handeln, aber nicht um Modellwarteschlangen. Wenn eine Modellwarteschlange angegeben wird, schlägt der Aufruf zum Öffnen dieser Warteschlange mit Ursachencode MQRC_Q_TYPE_ERROR fehl. Dies verhindert jedoch nicht, dass andere Warteschlangen in der Liste erfolgreich geöffnet werden.
4. Die Beendigungscode- und Ursachencodeparameter werden wie folgt gesetzt:
   - Wenn die Operationen zum Öffnen für die Warteschlangen in der Verteilerliste alle erfolgreich sind oder alle auf dieselbe Weise fehlschlagen, werden die Beendigungscode- und Ursachencodeparameter auf Werte gesetzt, die das allgemeine Ergebnis beschreiben. Die MQRR-Antwortdatensätze (falls von der Anwendung bereitgestellt) werden in diesem Fall nicht gesetzt.

     Wenn beispielsweise jede Operation zum Öffnen erfolgreich ist, werden der Beendigungscode auf MQCC_OK und der Ursachencode auf MQRC_NONE gesetzt. Schlägt jede Operation zum Öffnen fehl, weil keine der Warteschlangen vorhanden ist, werden die Parameter auf MQCC_FAILED und MQRC_UNKNOWN_OBJECT_NAME gesetzt.
   - Wenn die Operationen zum Öffnen für die Warteschlangen in der Verteilerliste nicht alle erfolgreich sind oder nicht alle auf dieselbe Weise fehlschlagen:
     - Der Beendigungscodeparameter wird auf MQCC_WARNING gesetzt, wenn mindestens eine Operation zum Öffnen erfolgreich ist, und auf MQCC_FAILED, wenn alle fehlgeschlagen sind.
     - Der Ursachencodeparameter wird auf MQRC_MULTIPLE_REASONS gesetzt.
     - Die Antwortdatensätze (falls von der Anwendung bereitgestellt) werden für die Warteschlangen in der Verteilerliste auf die einzelnen Beendigungscodes und Ursachencodes gesetzt.
5. Nachdem eine Verteilerliste erfolgreich geöffnet wurde, kann die vom Aufruf in *Hobj* zurückgegebene Kennung in nachfolgenden MQPUT-Aufrufen zum Einreihen von Nachrichten in Warteschlangen und in einem MQCLOSE-Aufruf zum Beenden des Zugriffs auf die Verteilerliste verwendet werden. Die einzige gültige Option zum Schließen für eine Verteilerliste ist MQCO_NONE.

   Auch mit dem MQPUT1-Aufruf kann eine Nachricht in eine Verteilerliste gestellt werden; die MQOD-Struktur, die die Warteschlangen in der Liste definiert, wird in dem Aufruf als Parameter angegeben.

6. Bei der Überprüfung, ob die Anwendung die maximal zulässige Anzahl Kennungen überschritten hat, wird jedes erfolgreich geöffnete Ziel in der Verteilerliste als separate Kennung gezählt (siehe Warteschlangenmanagerattribut *MaxHandles*). Dies gilt auch dann, wenn mehrere Ziele in der Verteilerliste in dieselbe physische Warteschlange aufgelöst werden. Wenn der MQOPEN- oder MQPUT1-Aufruf für eine Verteilerliste dazu führen würde, dass die Anzahl der von der Anwendung genutzten Kennungen das in *MaxHandles* angegebene Maximum überschreitet, schlägt der Aufruf mit Ursachencode MQRC_HANDLE_NOT_AVAILABLE fehl.
7. Für jedes erfolgreich geöffnete Ziel wird der Wert des zugehörigen Attributs *OpenOutputCount* um eins erhöht. Wenn mehrere Ziele in der Verteilerliste in dieselbe physische Warteschlange aufgelöst werden, wird das Attribut *OpenOutputCount* für diese Warteschlange um die Anzahl der Ziele in der Verteilerliste erhöht, die in diese Warteschlange aufgelöst werden.
8. Eine Änderung der Warteschlangendefinitionen, die zur Folge hat, dass eine Kennung ungültig wird, wenn die Warteschlangen einzeln geöffnet werden (z. B. eine Änderung im Auflösungspfad), führt nicht dazu, dass die Verteilerlistenkennung ungültig wird. Sie führt jedoch zu einem Fehler bei der betreffenden Warteschlange, wenn die Verteilerlistenkennung in einem nachfolgenden MQPUT-Aufruf verwendet wird.
9. Eine Verteilerliste kann auch nur ein einziges Ziel enthalten.

Ferne Warteschlangen

Die folgenden Hinweise gelten für die Verwendung von fernen Warteschlangen.

Eine ferne Warteschlange kann auf zwei Arten im Parameter *ObjDesc* dieses Aufrufs angegeben werden.

- Durch Angabe des Namens einer lokalen Definition der fernen Warteschlange im Feld *ObjectName*. In diesem Fall verweist das Feld *ObjectQMgrName* auf den lokalen Warteschlangenmanager, sodass Leerzeichen oder (in der Programmiersprache C) eine Nullzeichenfolge angegeben werden können.

  Bei der Sicherheitsprüfung, die der lokale Warteschlangenmanager durchführt, wird überprüft, ob der Benutzer berechtigt ist, die lokale Definition der fernen Warteschlange zu öffnen.
- Durch Angabe des Namens der fernen Warteschlange, so wie er dem fernen Warteschlangenmanager bekannt ist, im Feld *ObjectName*. In diesem Fall enthält das Feld *ObjectQMgrName* den Namen des fernen Warteschlangenmanagers.

  Bei der Sicherheitsprüfung, die der lokale Warteschlangenmanager durchführt, wird überprüft, ob der Benutzer berechtigt ist, Nachrichten an die Übertragungswarteschlange zu senden, die sich aus der Namensauflösung ergibt.

In beiden Fällen gilt:

- Es werden keine Nachrichten vom lokalen Warteschlangenmanager an den fernen Warteschlangenmanager gesendet, um zu überprüfen, ob der Benutzer berechtigt ist, Nachrichten in die Warteschlange zu stellen.
- Wenn eine Nachricht beim fernen Warteschlangenmanager eingeht, kann dieser die Nachricht zurückweisen, weil der Benutzer, von dem sie stammt, nicht berechtigt ist.

Weitere Informationen finden Sie in den Beschreibungen der Felder *ObjectName* und *ObjectQMgrName* im Abschnitt „MQOD - Objektdeskriptor” auf Seite 463.

Objekte

Sicherheit

Die folgenden Hinweise beziehen sich auf Sicherheitsaspekte bei der Verwendung des Aufrufs MQOPEN.

Wenn ein MQOPEN-Aufruf ausgegeben wird, führt der Warteschlangenmanager Sicherheitsprüfungen durch, um festzustellen, ob die Benutzer-ID, unter der die Anwendung ausgeführt wird, über die richtige Berechtigungsstufe verfügt; erst danach wird der Zugriff erlaubt. Die Berechtigungsprüfung wird für den

Namen des zu öffnenden Objekts durchgeführt und nicht für den oder die Namen, die sich aus der Auflösung eines Namens ergeben.

Wenn das zu öffnende Objekt eine Aliaswarteschlange ist, die auf ein Themenobjekt verweist, führt der Warteschlangenmanager eine Sicherheitsprüfung für den Namen der Aliaswarteschlange durch, bevor er eine Sicherheitsprüfung für das Thema durchführt, als wäre das Themenobjekt direkt verwendet worden.

Wenn das zu öffnende Objekt ein Themenobjekt ist, egal ob dabei nur das Feld *ObjectName* oder auch das Feld *ObjectString* (mit oder ohne einen zugrunde liegenden *ObjectName*) angegeben wird, führt der Warteschlangenmanager die Sicherheitsprüfung wie folgt durch: Er verwendet die entstehende Themenzeichenfolge, die dem im Feld *ObjectName* angegebenen Themenobjekt entnommen wird, und verkettet sie, falls erforderlich, mit der im Feld *ObjectString* angegebenen Zeichenfolge; anschließend sucht er das nächstgelegene Themenobjekt an oder über dem entsprechenden Punkt in der Themenstruktur und führt für dieses Objekt die Sicherheitsprüfung durch. Dabei handelt es sich möglicherweise nicht um dasselbe Themenobjekt, das im Feld *ObjectName* angegeben wurde.

Wenn das zu öffnende Objekt eine Modellwarteschlange ist, führt der Warteschlangenmanager sowohl für den Namen der Modellwarteschlange als auch für den Namen der dynamischen Warteschlange, die erstellt wird, eine vollständige Sicherheitsprüfung durch. Wird die erstellte dynamische Warteschlange anschließend explizit geöffnet, erfolgt eine weitere Ressourcensicherheitsprüfung für den Namen der dynamischen Warteschlange.

Attribute

Die folgenden Hinweise beziehen sich auf Attribute.

Die Attribute eines Objekts können geändert werden, während eine Anwendung das Objekt geöffnet hat. In vielen Fällen bekommt die Anwendung davon nichts mit, aber bei bestimmten Attributen markiert der Warteschlangenmanager die Kennung als nicht mehr gültig. Dabei handelt es sich um die folgenden Attribute:

- Jedes Attribut, das sich auf die Namensauflösung des Objekts auswirkt. Dies gilt unabhängig von den verwendeten Optionen zum Öffnen und schließt Folgendes ein:
  - Eine Änderung des Attributs *BaseQName* für eine geöffnete Aliaswarteschlange.
  - Eine Änderung des Attributs *TargetType* für eine geöffnete Aliaswarteschlange.
  - Eine Änderung des Warteschlangenattributs *RemoteQName* oder *RemoteQMgrName* für jede Kennung, die für diese Warteschlange geöffnet ist, oder für eine Warteschlange, die über diese Definition als ein Warteschlangenmanager-Aliasname aufgelöst wird.
  - Eine Änderung, die dazu führt, dass eine gerade geöffnete Kennung für eine ferne Warteschlange in eine andere Übertragungswarteschlange aufgelöst wird oder dass die Auflösung vollständig fehlschlägt. Dies können beispielsweise folgende Änderungen sein:
    - Eine Änderung des Attributs *XmitQName* der lokalen Definition einer fernen Warteschlange, egal ob die Definition für eine Warteschlange oder für einen Warteschlangenmanager-Aliasnamen verwendet wird.
    - Nur z/OS: Eine Änderung des Wertes des WS-Manager-Attributs *IntraGroupQueuing* oder eine Änderung der Definition der gemeinsam genutzten Übertragungswarteschlange (SYSTEM.QSG.TRANSMIT.QUEUE), die vom IGQ-Agenten verwendet wird.

    Es gibt hierzu nur eine einzige Ausnahme: die Erstellung einer neuen Übertragungswarteschlange. Eine Kennung, die in diese Warteschlange aufgelöst worden wäre, sofern sie beim Öffnen der Kennung vorhanden gewesen wäre, aber stattdessen in die Standardübertragungswarteschlange aufgelöst wurde, wird nicht als ungültig markiert.
  - Eine Änderung des Warteschlangenmanagerattributs *DefXmitQName*. In diesem Fall werden alle offenen Kennungen, die in die zuvor genannte Warteschlange aufgelöst wurden (und dies nur deshalb, weil es sich um die Standardübertragungswarteschlange handelte), als ungültig markiert. Kennungen, die aus anderen Gründen in diese Warteschlange aufgelöst wurden, sind nicht betroffen.
- Das Warteschlangenattribut *Shareability*, wenn es zwei oder mehrere Kennungen gibt, die aktuell MQOO_INPUT_SHARED-Zugriff für diese Warteschlange oder für eine Warteschlange, die in diese War-

teschlange aufgelöst wird, ermöglichen. In diesem Fall werden *alle* Kennungen, die für diese Warteschlange oder für eine Warteschlange, in die diese Warteschlange aufgelöst wird, als ungültig markiert, und das unabhängig von den Optionen zum Öffnen.

Unter z/OS werden die oben beschriebenen Kennungen als ungültig markiert, wenn eine oder mehrere Kennungen aktuell MQOO_INPUT_SHARED- oder MQOO_INPUT_EXCLUSIVE-Zugriff auf die Warteschlange ermöglichen.

- Das Warteschlangenattribut *Usage* für alle Kennungen, die für diese Warteschlange oder eine Warteschlange, die in diese Warteschlange aufgelöst wird, geöffnet sind, und das unabhängig von den Optionen zum Öffnen.

Wenn eine Kennung als ungültig markiert ist, schlagen alle nachfolgenden Aufrufe (außer MQCLOSE), die diese Kennung verwenden, mit Ursachencode MQRC_OBJECT_CHANGED fehl. Die Anwendung muss einen MQCLOSE-Aufruf (mit der ursprünglichen Kennung) ausgeben und die Warteschlange dann erneut öffnen. Nicht festgeschriebene Aktualisierungen für die alte Kennung aus vorherigen erfolgreichen Aufrufen können trotzdem festgeschrieben oder zurückgesetzt werden, so wie von der Anwendungslogik gefordert.

Wenn die Änderung eines Attributs zu einer solchen Situation führt, verwenden Sie eine spezielle Version des Aufrufs, um die Aktion zu erzwingen.

C-Aufruf

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
       &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Nachricht einreihen

Der MQPUT-Aufruf reiht eine Nachricht in eine Warteschlange oder Verteilerliste ein oder ordnet sie einem Thema zu. Die Warteschlange, die Verteilerliste oder das Thema muss bereits geöffnet sein.

Syntax

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Für CICS-Anwendungen unter z/OS und Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und folgender Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Hobj
Typ: MQHOBJ - Eingabe

Diese Kennung steht für die Warteschlange, in welche die Nachricht gestellt wird, oder für das Thema, zu dem die Nachricht veröffentlicht wird. Der Wert von *Hobj* wurde von einem früheren MQOPEN-Aufruf zurückgegeben, in dem die Option MQOO_OUTPUT angegeben war.

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Diese Struktur beschreibt die Attribute der gesendeten Nachricht und erhält nach ausgeführter Einreihungsanforderung Informationen über die Nachricht. Details siehe „MQMD - Nachrichtendeskriptor" auf Seite 399.

Wenn die Anwendung einen MQMD der Version 1 bereitstellt, kann den Nachrichtendaten eine MQMDE-Struktur vorangestellt werden, um Werte für die Felder anzugeben, die im MQMD der Version 2, aber nicht in Version 1 vorhanden sind. Das Feld *Format* im MQMD muss auf MQFMT_MD_EXTENSION festgelegt sein um anzugeben, dass eine MQMDE vorhanden ist. Weitere Informationen finden Sie unter „MQMDE – Nachrichtendeskriptorerweiterung" auf Seite 453.

Die Anwendung muss keine MQMD-Struktur bereitstellen, wenn eine gültige Nachrichtenkennung im Feld *OriginalMsgHandle* oder *NewMsgHandle* der MQPMO-Struktur bereitgestellt wird. Wenn in einem dieser Felder nichts bereitgestellt wird, wird als Deskriptor der Nachricht der Deskriptor übernommen, der den Nachrichtenkennungen zugeordnet ist.

Wenn Sie API-Exits verwenden oder die Verwendung planen, wird empfohlen, explizit eine MQMD-Struktur bereitzustellen und nicht die Nachrichtendeskriptoren zu verwenden, die den Nachrichtenkennungen zugeordnet sind. Der Grund ist, dass der dem MQPUT- oder MQPUT1-Aufruf zugeordnete API-Exit nicht in der Lage ist zu bestimmen, welche MQMD-Werte vom Warteschlangenmanager verwendet werden, um die MQPUT- oder MQPUT1-Anforderung abzuschließen.

PutMsgOpts
Typ: MQPMO - Ein-/Ausgabe

Details siehe „MQPMO - Nachrichteneinreihungsoptionen" auf Seite 484.

BufferLength
Typ: MQLONG - Eingabe

Die Länge der Nachricht in *Buffer*. Null ist gültig und zeigt an, dass die Nachricht keine Anwendungsdaten enthält. Die Obergrenze für *BufferLength* hängt von mehreren Faktoren ab.

- Wenn die Zieladresse eine lokale Warteschlange ist oder in eine lokale Warteschlange aufgelöst wird, hängt die Obergrenze davon ab, ob:
  - Der lokale Warteschlangenmanager unterstützt Segmentierung.
  - Die sendende Anwendung gibt das Flag an, das dem Warteschlangenmanager ermöglicht, die Nachricht zu segmentieren. Dieses Flag ist MQMF_SEGMENTATION_ALLOWED und kann entweder in einem MQMD der Version 2 oder in einer MQMDE angegeben werden, die mit einem MQMD der Version 1 verwendet wird.

  Wenn beide Bedingungen erfüllt sind, darf *BufferLength* nicht 999 999 999 minus dem Wert des Feldes *Offset* im MQMD überschreiten. Die maximale Länge der logischen Nachricht, die eingereiht werden kann, beträgt daher 999 999 999 Byte (wenn *Offset* null ist). Allerdings können die Ressourcenbeschränkungen durch das Betriebssystem oder durch die Umgebung, in der die Anwendung ausgeführt wird, zu einer niedrigeren Obergrenze führen.

  Wenn eine oder beide oben genannten Bedingungen nicht erfüllt sind, darf *BufferLength* nicht den kleineren Wert des Warteschlangenattributs *MaxMsgLength* bzw. des WS-Manager-Attributs *MaxMsgLength* überschreiten.
- Wenn das Ziel einer fernen Warteschlange ist oder als ferne Warteschlange aufgelöst wird, gelten die Bedingungen für lokale Warteschlangen, *jedoch an jedem Warteschlangenmanager, den die Nachricht passieren muss, um die Zielwarteschlange zu erreichen*; insbesondere:
  1. Die lokale Übertragungswarteschlange für die temporäre Speicherung der Nachricht beim lokalen Warteschlangenmanager

2. Temporäre Übertragungswarteschlangen (falls vorhanden) für die Speicherung der Nachricht bei Warteschlangenmanagern zwischen dem lokalen und dem Zielwarteschlangenmanager
3. Die Zielwarteschlange beim Zielwarteschlangenmanager

Die längste Nachricht, die eingereiht werden kann, wird deshalb durch die Warteschlangen und Warteschlangenmanager bestimmt, die den weitestgehenden Einschränkungen unterworfen sind.

Wenn sich eine Nachricht in einer Übertragungswarteschlange befindet, enthalten die Nachrichtendaten zusätzliche Informationen, wodurch sich die Menge der Anwendungsdaten, die transportiert werden können, reduziert. In diesem Fall subtrahieren Sie MQ_MSG_HEADER_LENGTH-Werte von den *MaxMsgLength*-Werten der Übertragungswarteschlangen, wenn Sie den Grenzwert für *BufferLength* ermitteln.

Anmerkung: Nur die Nichterfüllung der Bedingung 1 kann synchron diagnostiziert werden (mit Ursachencode MQRC_MSG_TOO_BIG_FOR_Q oder MQRC_MSG_TOO_BIG_FOR_Q_MGR), wenn die Nachricht eingereiht wird. Wenn Bedingung 2 oder 3 nicht erfüllt wird, wird die Nachricht in eine Warteschlange für nicht zustellbare Nachrichten umgeleitet, entweder bei einem zwischengeschalteten Warteschlangenmanager oder beim Zielwarteschlangenmanager. Wenn dies eintritt, wird eine Berichtsnachricht generiert, falls vom Absender angefordert.

Buffer
Typ: MQBYTExBufferLength - Eingabe

Dies ist ein Puffer, der die zu sendenden Anwendungsdaten enthält. Der Puffer muss an einem Grenzwert ausgerichtet sein, der der Spezifik der Daten in der Nachricht entspricht. 4-Byte-Ausrichtung ist für die meisten Nachrichten geeignet (einschließlich Nachrichten mit WebSphere MQ-Headerstrukturen), aber manche Nachrichten erfordern eine stringentere Ausrichtung. Zum Beispiel kann eine Nachricht, die eine binäre Ganzzahl von 64 Bit enthält, eine 8-Bit-Ausrichtung erfordern.

Wenn *Buffer* Zeichen- oder numerische Daten enthält, setzen Sie die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* auf die Werte, die den Daten entsprechen. Dies ermöglicht es dem Empfänger der Nachricht, die Daten in den verwendeten Zeichensatz und die verwendete Codierung zu konvertieren.

Anmerkung: Alle anderen Parameter beim MQPUT-Aufruf müssen dem Zeichensatz und der Codierung des lokalen Warteschlangenmanagers entsprechen (durch das Warteschlangenmanagerattribut *CodedCharSetId* und durch MQENC_NATIVE angegeben).

In der Programmiersprache C ist der Parameter als pointer-to-void deklariert; als Parameter kann die Adresse eines jeden beliebigen Datentyps angegeben werden.

Wenn der Parameter *BufferLength* null ist, wird *Buffer* nicht referenziert. Die Adresse, die von Programmen übergeben wird, die in C oder in System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Nachrichtengruppe nicht vollständig

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Logische Nachricht nicht vollständig

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889') Inkonsistente Persistenzspezifikation

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801') Nachrichtenpriorität überschreitet unterstützten Maximalwert

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838') Berichtsoption(en) im Nachrichtendeskriptor nicht erkannt

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ALIAS_TARGTYPE_CHANGED
(2480, X'09B0') Subskriptionszieltyp hat sich vom Warteschlangen- zum Themenobjekt geändert

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BACKED_OUT
(2003, X'7D3') Arbeitseinheit zurückgesetzt

MQRC_BUFFER_ERROR
(2004, X'7D4') Pufferparameter ungültig

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT oder MQCMIT wurde unterbrochen und die Verbindungswiederholung kann kein definitives Ergebnis wiederherstellen

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CFGR_ERROR
(2416, X'970') PCF-Gruppenparameterstruktur MQCFGR in den Nachrichtendaten ungültig

MQRC_CFH_ERROR
(2235, X'8BB') PCF-Headerstruktur ungültig

MQRC_CFIF_ERROR
(2414, X'96E') PCF-Ganzzahlfilterparameterstruktur in den Nachrichtendaten ungültig

MQRC_CFIL_ERROR
(2236, X'8BC') PCF-Ganzzahllistenparameterstruktur oder PCIF*64-Ganzzahllistenparameterstruktur ungültig

MQRC_CFIN_ERROR
(2237, X'8BD') PCF-Ganzzahlparameterstruktur oder PCIF*64-Ganzzahlparameterstruktur ungültig

MQRC_CFSF_ERROR
(2415, X'96F') PCF-Zeichenfolgefilterparameterstruktur in den Nachrichtendaten ungültig

MQRC_CFSL_ERROR
(2238, X'8BE') PCF-Zeichenfolgelistenparameterstruktur ungültig

MQRC_CFST_ERROR
(2239, X'8BF') PCF-Zeichenfolgeparameterstruktur ungültig

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Exit für Clusterauslastung fehlgeschlagen.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Clusternamensauflösung fehlgeschlagen.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Clusterressourcenfehler.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') COD-Berichtsoption für XCF-Warteschlange ungültig

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CONTENT_ERROR
2554 (X'09FA') Nachrichteninhalt konnte nicht analysiert werden um zu ermitteln, ob die Nachricht einem Subskribenten mit erweitertem Nachrichtenselektor übergeben werden soll

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831') Referenzierte Warteschlangenkennung speichert keinen Kontext

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832') Kein Kontext für die angegebene Warteschlangenkennung vorhanden.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Parameter Datenlänge ungültig.

MQRC_DH_ERROR
(2135, X'857') Verteilungs-Headerstruktur ungültig

MQRC_DLH_ERROR
(2141, X'85D') Headerstruktur für nicht zustellbare Nachrichten ungültig

MQRC_EPH_ERROR
(2420, X'974') Eingebettete PCF-Struktur ungültig

MQRC_EXPIRY_ERROR
(2013, X'7DD') Ablaufzeit ungültig

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Rückkopplungscode ungültig

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Konflikt mit globaler Arbeitseinheit.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Gruppen-ID ungültig

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Kennung für globale Arbeitseinheit belegt.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HEADER_ERROR
(2142, X'85E') MQ-Headerstruktur ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_IIH_ERROR
(2148, X'864') Headerstruktur für IMS-Informationen ungültig

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Nachrichtengruppe nicht vollständig

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Logische Nachricht nicht vollständig

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889') Inkonsistente Persistenzspezifikation

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Spezifikation für Arbeitseinheit inkonsistent.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') Globale Arbeitseinheit im Konflikt mit lokaler Arbeitseinheit.

MQRC_MD_ERROR
(2026, X'7EA') Nachrichtendeskriptor ungültig

MQRC_MDE_ERROR
(2248, X'8C8') Nachrichtendeskriptorerweiterung ungültig

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Fehlende Warteschlange für zu beantwortende Nachrichten oder MQPMO_SUPPRESS_REPLYTO wurde verwendet

MQRC_MISSING_WIH
(2332, X'91C') Nachrichtendaten beginnen nicht mit MQWIH

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Nachrichtenflags ungültig

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Nachrichtenfolgenummer ungültig

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Nachrichtenlänge überschreitet den Maximalwert für die Warteschlange.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Die Nachrichtenlänge überschreitet den maximalen Wert des Warteschlangenmanagers.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Nachrichtentyp im Nachrichtendeskriptor ungültig

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Keine Zielwarteschlangen verfügbar

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Warteschlange nicht für Ausgabe geöffnet

MQRC_NOT_OPEN_FOR_PASS_ALL
(2093, X'82D') Warteschlange nicht für Übergabe des gesamten Kontextes geöffnet

MQRC_NOT_OPEN_FOR_PASS_IDENT
(2094, X'82E') Warteschlange nicht für Übergabe des Identitätskontextes geöffnet

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') Warteschlange nicht für Festlegung des gesamten Kontextes geöffnet

MQRC_NOT_OPEN_FOR_SET_IDENT
(2096, X'830') Warteschlange nicht für Festlegung des Identitätskontextes geöffnet

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OFFSET_ERROR
(2251, X'8CB') Nachrichtensegmentoffset ungültig

MQRC_OPEN_FAILED
(2137, X'859') Objekt nicht erfolgreich geöffnet

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Ursprüngliche Länge ungültig

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_PAGESET_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_PCF_ERROR
(2149, X'865') PCF-Strukturen ungültig

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistenz ungültig

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800') Die Warteschlange unterstützt keine persistenten Nachrichten.

MQRC_PMO_ERROR
(2173, X'87D') Put-Message-Optionsstruktur ungültig

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Put-Message-Datensatzflags ungültig

MQRC_PRIORITY_ERROR
(2050, X'802') Nachrichtenpriorität ungültig

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') Die Veröffentlichung wurde an keinen der Subskribenten übermittelt.

MQRC_PUT_INHIBITED
(2051, X'803') Put-Aufrufe für diese Warteschlange, für die Warteschlange, in die diese Warteschlange aufgelöst wird oder für das Thema unterdrückt

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Nachrichteneinreihungssätze ungültig.

MQRC_PUT_NOT_RETAINED
(2479, X'09AF') Veröffentlichung konnte nicht beibehalten werden

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_FULL
(2053, X'805') Die Warteschlange enthält bereits die maximale Anzahl von Nachrichten.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') Auf dem Datenträger ist für die Warteschlange kein Speicherplatz verfügbar.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Nach der Wiederverbindung ist ein Fehler aufgetreten, der die Kennungen für eine wiederverbindbare Verbindung wiedereingesetzt hat

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Anzahl vorhandener Datensätze ungültig.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D) Berichtsoptionen in Nachrichtendeskriptor ungültig.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Antwortdatensätze ungültig.

MQRC_RFH_ERROR
(2334, X'91E') MQRFH- oder MQRFH2-Struktur ungültig

MQRC_RMH_ERROR
(2220, X'8AC') Headerstruktur der Referenznachricht ungültig

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') Länge der Daten im Nachrichtensegment ist null.

MQRC_SEGMENTS_NOT_SUPPORTED
(2365, X'93D') Segmente nicht unterstützt

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Ein möglicher Subskribent für die Veröffentlichung existiert, aber der Warteschlangenmanager kann nicht überprüfen, ob die Veröffentlichung an den Subskribenten gesendet werden soll

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Aufruf vom Exit für Clusterauslastung zurückgewiesen.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839') Speicherklassenfehler

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') In der aktuellen Arbeitseinheit können keine weiteren Nachrichten verarbeitet werden.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') Unterstützung für Synchronisationspunkt nicht verfügbar.

MQRC_TM_ERROR
(2265, X'8D9') Auslösenachrichtstruktur ungültig

MQRC_TMC_ERROR
(2191, X'88F') Zeichenauslösenachrichtstruktur ungültig

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Eintragung in globale Arbeitseinheit fehlgeschlagen.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') Mischung von Aufrufen für Arbeitseinheiten wird nicht unterstützt.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') Arbeitseinheit nicht zur Verwendung durch den Warteschlangenmanager verfügbar.

MQRC_WIH_ERROR
(2333, X'91D') MQWIH-Struktur ungültig

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Falsche MQMD-Version bereitgestellt.

MQRC_XQH_ERROR
(2260, X'8D4') Headerstruktur der Übertragungswarteschlange ungültig

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung von Themen

1. Die folgenden Hinweise gelten für die Verwendung von Themen:

   a. Bei Verwendung von MQPUT für das Veröffentlichen von Nachrichten zu einem Thema gilt: Wenn einem oder mehreren Subskribenten zu dem Thema die Veröffentlichung wegen eines Problems mit Warteschlange dieser Teilnehmerliste für Teilnehmerberechtigungen (wenn sie zum Beispiel voll ist) nicht übermittelt werden kann, sind der beim MQPUT-Aufruf zurückgegebene Ursachencode und das Übermittlungsverhalten abhängig von der Einstellung der Attribute PMSGDLV oder NPMSGDLV zum THEMA. Die Übergabe einer Veröffentlichung an die Warteschlange für nicht zustellbare Nachrichten bei Angabe von MQRO_DEAD_LETTER_Q oder das Verwerfen der Nachricht bei Angabe von MQRO_DISCARD_MSG gilt als erfolgreiche Übergabe der Nachricht. Wenn keine Veröffentlichungen übergeben werden, wird MQPUT mit MQRC_PUBLICATION_FAILURE zurückgegeben. Dies ist unter den folgenden Bedingungen der Fall:

      - Eine Nachricht wird zu einem THEMA veröffentlicht, wobei (je nach der Persistenz der Nachricht) PMSGDLV oder NPMSGDLV auf ALL gesetzt ist und eine (permanente oder nicht permanente) Subskription eine Warteschlange hat, die die Veröffentlichung nicht erhalten kann.
      - Eine Nachricht wird zu einem THEMA veröffentlicht, wobei (je nach der Persistenz der Nachricht) PMSGDLV oder NPMSGDLV auf ALLDUR gesetzt ist und eine permanente Subskription eine Warteschlange hat, die die Veröffentlichung nicht erhalten kann.

      MQPUT kann mit MQRC_NONE zurückgegeben werden, auch wenn Veröffentlichungen an einige Subskribenten in den folgenden Fällen nicht übergeben werden konnten:

      - Eine Nachricht wird zu einem THEMA veröffentlicht, wobei (je nach der Persistenz der Nachricht) PMSGDLV oder NPMSGDLV auf ALLAVAIL gesetzt ist und irgendeine permanente oder nicht permanente Subskription eine Warteschlange hat, die die Veröffentlichung nicht erhalten kann.
      - Eine Nachricht wird zu einem Thema veröffentlicht, wobei (je nach der Persistenz der Nachricht) PMSGDLV oder NPMSGDLV auf ALLDUR gesetzt ist und eine nicht permanente Subskription eine Warteschlange hat, die die Veröffentlichung nicht erhalten kann.

      Mit dem Themenattribut USEDLQ können Sie festlegen, ob die Warteschlange für nicht zustellbare Nachrichten verwendet werden soll, wenn Veröffentlichungsnachrichten nicht an ihre korrekte Warteschlange für Subskribenten übermittelt werden können. Weitere Informationen zur Verwendung von USEDLQ finden Sie unter DEFINE TOPIC.

b. Wenn es keine Subskribenten zum verwendeten Thema gibt, wird die veröffentlichte Nachricht an keine Warteschlange gesendet, sondern wird verworfen. Unabhängig davon, ob die Nachricht persistent oder nicht persistent ist und ob sie eine Ablaufzeit hat oder nicht, wird sie verworfen, wenn es keine Subskribenten gibt. Eine Ausnahme gilt, wenn die Nachricht beibehalten werden soll: In diesem Fall wird sie zwar nicht an Subskribentenwarteschlangen gesendet, aber zum an neue Subskriptionen zu übermittelnden Thema oder für Subskribenten, die mit MQSUBRQ ständige Veröffentlichungen anfordern, gespeichert.

MQPUT und MQPUT1

Entsprechend den Anforderungen können Sie den MQPUT- oder den MQPUT1-Aufruf verwenden, um Nachrichten in eine Warteschlange einzureihen.

- Verwenden Sie den MQPUT-Aufruf, um mehrere Nachrichten in *dieselbe* Warteschlange zu platzieren.

  Zuerst wird ein MQOPEN-Aufruf mit Angabe der Option MQOO_OUTPUT ausgegeben, gefolgt von mindestens einer MQPUT-Anforderung zum Hinzufügen von Nachrichten zur Warteschlange. Zuletzt wird die Warteschlange mit einem MQCLOSE-Aufruf geschlossen. Dieses Vorgehen ermöglicht eine bessere Leistung als wiederholte MQPUT1-Aufrufe.
- Verwenden Sie den MQPUT1-Aufruf, um nur *eine* Nachricht in die Warteschlange einzureihen.

  Dieser Aufruf bindet die MQOPEN-, MQPUT- und MQCLOSE-Aufrufe in einen einzigen Aufruf ein, wodurch die Anzahl der auszugebenden Aufrufe minimiert wird.

Zielwarteschlangen

Die folgenden Hinweise gelten für die Verwendung von Zielwarteschlangen:

1. Wenn eine Anwendung eine Folge von Nachrichten in dieselbe Warteschlange einreiht, ohne Nachrichtengruppen zu verwenden,wird die Reihenfolge dieser Nachrichten beibehalten, wenn die angegebenen Bedingungen erfüllt sind. Einige Bedingungen gelten sowohl für lokale als auch für ferne Zielwarteschlangen, andere nur für ferne Zielwarteschlangen.

   Bedingungen, die für lokale und ferne Zielwarteschlangen gelten

   - Alle MQPUT-Aufrufe erfolgen innerhalb derselben Arbeitseinheit oder keiner von ihnen erfolgt innerhalb einer Arbeitseinheit.

     Beachten Sie, dass beim Einreihen von Nachrichten in eine bestimmte Warteschlange in einer einzelnen Arbeitseinheit möglicherweise Nachrichten von anderen Anwendungen in die Folge von Nachrichten in der Warteschlange eingefügt werden.
   - Alle MQPUT-Aufrufe erfolgen mit der gleichen Objektkennung *Hobj*.

     In einigen Umgebungen wird die Nachrichtenreihenfolge auch beibehalten, wenn verschiedene Objektkennungen verwendet werden, wenn die Aufrufe von derselben Anwendung kommen. Die Bedeutung von *dieselbe Anwendung* wird von der Umgebung bestimmt.

     – Unter z/OS ist die Anwendung:

       - Bei CICS die CICS-Task
       - Bei IMS die Task
       - Bei z/OS-Stapel die Task

     – Unter IBM i ist die Anwendung der Job

     – Auf Windows- und UNIX-Systemen ist die Anwendung der Thread
   - Die Nachrichten haben alle dieselbe Priorität.
   - Die Nachrichten werden nicht mit MQOO_BIND_NOT_FIXED (oder mit MQOO_BIND_AS_Q_DEF, wenn das Warteschlangenattribut "DefBind" den Wert MQBND_BIND_NOT_FIXED hat) in eine Clusterwarteschlange eingereiht.

   Zusätzliche Bedingungen, die für ferne Zielwarteschlangen gelten

- Es gibt nur einen Pfad vom sendenden Warteschlangenmanager zum Zielwarteschlangenmanager.

  Wenn einige Nachrichten in der Folge über einen anderen Pfad übermittelt werden (z. B. wegen Rekonfiguration, Datenverkehrlastausgleich oder einer auf Nachrichtengröße basierenden Pfadauswahl), kann die Reihenfolge der Nachrichten beim Zielwarteschlangenmanager nicht garantiert werden.
- Nachrichten werden bei den sendenden, temporären und Zielwarteschlangenmanagern nicht vorübergehend in Warteschlangen für nicht zustellbare Nachrichten gestellt.

  Wird eine oder mehr der Nachrichten zeitweilig in eine Warteschlange für nicht zustellbare Nachrichten eingereiht (zum Beispiel weil eine Übertragungswarteschlange oder die Zielwarteschlange vorübergehend voll ist), können die Nachrichten in der Zielwarteschlange in der falschen Reihenfolge eintreffen.
- Die Nachrichten sind entweder alle persistent oder alle nicht persistent.

  Wenn bei einem Kanal auf der Route zwischen dem sendenden und dem Ziel-WS-Manager das Attribut *NonPersistentMsgSpeed* auf MQNPMS_FAST festgelegt ist, können nicht persistente Nachrichten vor persistente Nachrichten springen, dadurch wird die Reihenfolge von persistenten Nachrichten und nicht persistenten Nachrichten geändert. Dabei bleibt jedoch die Reihenfolge zwischen den persistenten und die Reihenfolge zwischen den nicht persistenten Nachrichten unverändert.

Wenn diese Bedingungen nicht erfüllt sind, können Sie Nachrichtengruppen verwenden, um die Nachrichtenreihenfolge beizubehalten. Dies erfordert, dass sowohl die sendenden als auch die empfangenden Anwendungen die Nachrichtengruppierung unterstützen. Weitere Informationen zu Nachrichtengruppen finden Sie in den folgenden Abschnitten:

- MQMD - MsgFlags-Feld
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Verteilerlisten

Die folgenden Hinweise gelten für die Verwendung von Verteilerlisten.

Verteilerlisten werden in folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients, die mit diesen Systemen verbunden sind.

1. Sie können Nachrichten in Verteilerlisten mit MQPMO der Version 1 oder der Version 2 einreihen. Wenn Sie MQPMO der Version 1 (oder MQPMO der Version 2 mit *RecsPresent* gleich null) verwenden, kann die Anwendung keine Nachrichteneinreihungs- oder Antwortdatensätze bereitstellen. Sie können nicht die Warteschlangen ermitteln, bei denen Fehler auftreten, wenn die Nachricht erfolgreich an einige Warteschlangen in der Verteilerliste gesendet wird und an andere nicht.

   Wenn die Anwendung Nachrichteneinreihungssätze oder Antwortdatensätze bereitstellt, legen Sie das Feld *Version* auf MQPMO_VERSION_2 fest.

   Sie können auch MQPMO der Version 2 verwenden, um Nachrichten an eine einzelne Warteschlange zu senden, die nicht in einer Verteilerliste enthalten ist, wenn *RecsPresent* gleich null ist.
2. Die Beendigungscode- und Ursachencodeparameter werden wie folgt gesetzt:

   - Wenn die Put-Operationen zum Einreihen von Nachrichten in die Warteschlangen in der Verteilerliste alle erfolgreich sind oder alle auf die gleiche Weise fehlschlagen, werden die Beendigungscode- und Ursachencodeparameter auf Werte gesetzt, die das generelle Ergebnis beschreiben. Die MQRR-Antwortdatensätze (falls von der Anwendung bereitgestellt) werden in diesem Fall nicht gesetzt.

     Wenn beispielsweise jede Put-Operation erfolgreich ist, werden der Beendigungscode auf MQCC_OK und der Ursachencode auf MQRC_NONE gesetzt. Schlägt jede Put-Operation fehl, weil alle Warteschlangen für Einreihungen gesperrt sind, werden die Parameter auf MQCC_FAILED und MQRC_PUT_INHIBITED gesetzt.

- Wenn die Einreihungen in die Warteschlangen in der Verteilerliste nicht alle auf dieselbe Weise erfolgreich sind oder fehlschlagen:
  - Der Beendigungscodeparameter wird auf MQCC_WARNING gesetzt, wenn mindestens eine Put-Operation erfolgreich ist, und auf MQCC_FAILED, wenn alle fehlgeschlagen sind.
  - Der Ursachencodeparameter wird auf MQRC_MULTIPLE_REASONS gesetzt.
  - Die Antwortdatensätze (falls von der Anwendung bereitgestellt) werden für die Warteschlangen in der Verteilerliste auf die einzelnen Beendigungscodes und Ursachencodes gesetzt.

  Wenn das Einreihen in ein Ziel fehlschlägt, weil eine Operation zum Öffnen dieses Ziels fehlgeschlagen ist, werden die Felder im Antwortdatensatz auf MQCC_FAILED und MQRC_OPEN_FAILED gesetzt. Dieses Ziel wird in *InvalidDestCount* aufgenommen.

3. Wenn ein Ziel in der Verteilerliste in eine lokale Warteschlange aufgelöst wird, wird die Nachricht in Normalform (also nicht als Verteilerlistennachricht) in diese Warteschlange eingereiht. Wenn mehr als ein Ziel in dieselbe lokale Warteschlange aufgelöst wird, wird für jedes der Ziele eine Nachricht in die Warteschlange eingereiht.

   Wenn ein Ziel in der Verteilerliste als ferne Warteschlange aufgelöst wird, wird eine Nachricht in die entsprechende Übertragungswarteschlange gestellt. Wenn mehrere Ziele in dieselbe Übertragungswarteschlange aufgelöst werden, kann eine einzelne Verteilerlistennachricht mit diesen Zielen in die Übertragungswarteschlange eingefügt werden, auch wenn diese Ziele in der Liste, die von der Anwendung bereitgestellt wurde, nicht benachbart waren. Dies ist allerdings nur möglich, wenn die Übertragungswarteschlange Verteilerlistennachrichten unterstützt (siehe DistLists).

   Wenn die Übertragungswarteschlange keine Verteilerlisten unterstützt, wird für jedes Ziel, das die Übertragungswarteschlange nutzt, eine Kopie der Nachricht in Normalform abgestellt.

   Wenn eine Verteilerliste mit den Anwendungsnachrichtendaten zu groß für eine Übertragungswarteschlange ist, wird die Verteilerliste in kleinere Verteilerlistennachrichten mit jeweils weniger Zielen aufgeteilt. Wenn die Anwendungsnachrichtendaten nur gerade noch in die Warteschlange passen, können Verteilerlistennachrichten überhaupt nicht verwendet werden und der Warteschlangenmanager erstellt für jedes Ziel, das diese Übertragungswarteschlange verwendet, eine Kopie der Nachricht in Normalform.

   Wenn verschiedene Ziele unterschiedliche Nachrichtenpriorität oder Nachrichtenpersistenz haben (dies kann auftreten, wenn die Anwendung MQPRI_PRIORITY_AS_Q_DEF oder MQPER_PERSISTENCE_AS_Q_DEF angibt), werden die Nachrichten nicht in derselben Verteilerlistennachricht gehalten. Stattdessen generiert der Warteschlangenmanager so viele Verteilerlistennachrichten wie erforderlich, um die verschiedenen Prioritäts- und Persistenzwerte aufzunehmen.

4. Das Einreihen in eine Verteilerliste kann folgendes Ergebnis haben:
   - Eine einzelne Verteilerlistennachricht oder
   - Mehrere kleinere Verteilerlistennachrichten oder
   - Eine Mischung aus Verteilerlistennachrichten und normalen Nachrichten oder
   - Nur normale Nachrichten.

   Welche der oben genannten Möglichkeiten eintritt, hängt davon ab, ob:

   - Die Ziele in der Liste lokal, fern oder eine Mischung daraus sind
   - Die Ziele haben die gleiche Nachrichtenpriorität und -persistenz.
   - Die Übertragungswarteschlangen Verteilerlistennachrichten aufnehmen können
   - Die maximalen Nachrichtenlängen der Übertragungswarteschlangen sind groß genug, um die Nachricht in Verteilungslistenform aufzunehmen.

   Unabhängig vom tatsächlichen Resultat zählt jedoch jede resultierende *physische* Nachricht (das heißt, jede normale Nachricht oder Verteilerlistennachricht, die Ergebnis des Einreihens ist, in den folgenden Situationen als nur *eine* Nachricht:

   - Überprüfen, ob die Anwendung die zulässige maximale Anzahl Nachrichten in einer Arbeitseinheit überschritten hat (siehe Warteschlangenmanagerattribut *MaxUncommittedMsgs*.

- Überprüfen, ob die Auslösebedingungen erfüllt werden.
- Bei der Erhöhung der Warteschlangenlängen und der Prüfung, ob dadurch die maximale Länge der Warteschlange überschritten würde.

5. Eine Änderung der Warteschlangendefinitionen, die zur Folge hat, dass eine Kennung ungültig wird, wenn die Warteschlangen einzeln geöffnet werden (z. B. eine Änderung im Auflösungspfad), führt nicht dazu, dass die Verteilerlistenkennung ungültig wird. Sie führt jedoch zu einem Fehler bei der betreffenden Warteschlange, wenn die Verteilerlistenkennung in einem nachfolgenden MQPUT-Aufruf verwendet wird.

Kopfzeilen

Wenn eine Nachricht mit einer oder mehreren WebSphere MQ-Headerstrukturen zu Beginn der Anwendungsnachrichtendaten eingereiht wird, führt der Warteschlangenmanager bestimmte Überprüfungen der Headerstrukturen durch, um die Gültigkeit zu verifizieren. Wenn der Warteschlangenmanager einen Fehler feststellt, schlägt der Aufruf mit einem entsprechenden Ursachencode fehl. Die durchgeführten Überprüfungen hängen von den jeweiligen Strukturen ab, die vorhanden sind:

- Überprüfungen werden nur durchgeführt, wenn ein MQMD der Version 2 oder höher beim MQPUT- oder MQPUT1-Aufruf verwendet wird. Es werden keine Überprüfungen durchgeführt, wenn ein MQMD der Version 1 verwendet wird, selbst wenn eine MQMDE am Anfang der Nachrichtendaten vorhanden ist.
- Strukturen, die nicht vom lokalen Warteschlangenmanager unterstützt werden, und Strukturen, die auf den ersten MQDLH in der Nachricht folgen, werden nicht validiert.
- MQDH- und MQMDE-Strukturen werden vom Warteschlangenmanager vollständig validiert.
- Andere Strukturen werden vom Warteschlangenmanager teilweise validiert (nicht alle Felder werden überprüft).

Allgemeine Überprüfungen durch den Warteschlangenmanager sind folgende:

- Das Feld *StrucId* muss gültig sein.
- Das Feld *Version* muss gültig sein.
- Das Feld *StrucLength* muss einen Wert angeben, der groß genug ist, um die Struktur sowie Daten mit variabler Länge aufzunehmen, die Teil der Struktur sind.
- Das Feld *CodedCharSetId* darf nicht null oder ein negativer Wert sein, der ungültig ist (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR und MQCCSI_UNDEFINED sind in den meisten WebSphere MQ-Headerstrukturen *nicht* gültig).
- Der Parameter *BufferLength* des Aufrufs muss einen Wert angeben, der groß genug ist, um die Struktur aufzunehmen (die Struktur darf nicht über das Ende der Nachricht hinausgehen).

Zusätzlich zu allgemeinen Überprüfungen von Strukturen müssen die folgenden Bedingungen erfüllt werden:

- Die Summe der Längen aller Strukturen in einer PCF-Nachricht muss der durch den Parameter *BufferLength* im MQPUT- oder MQPUT1-Aufruf angegebenen Länge entsprechen. Eine PCF-Nachricht ist eine Nachricht, die den Formatnamen MQFMT_ADMIN, MQFMT_EVENT oder MQFMT_PCF hat.
- Eine WebSphere MQ-Struktur darf nicht abgeschnitten werden, ausgenommen folgende Situationen, bei denen gekürzte Strukturen zulässig sind:
  - Nachrichten, die Berichtsnachrichten sind
  - PCF-Nachrichten.
  - Nachrichten, die eine MQDLH-Struktur enthalten. (Strukturen *nach* dem ersten MQDLH können abgeschnitten werden; Strukturen, die dem MQDLH vorangehen, können es nicht).
- Eine WebSphere MQ-Struktur darf nicht auf zwei oder mehr Segmente aufgeteilt werden. Die Struktur muss vollständig in einem Segment enthalten sein.

Puffer

Für die Programmiersprache Visual Basic gilt Folgendes:

- Wenn die Größe des Parameters *Buffer* kleiner als die Länge ist, die im Parameter *BufferLength* angegeben ist, schlägt der Aufruf mit Ursachencode MQRC_BUFFER_LENGTH_ERROR fehl.
- Der Parameter *Buffer* wird als Typ `String` deklariert. Wenn es sich bei den Daten, die in die Warteschlange eingefügt werden sollen, nicht um Daten des Typs `String` handelt, verwenden Sie denMQPUTAny-Aufruf anstelle von MQPUT.

  Der MQPUTAny-Aufruf hat dieselben Parameter wie der MQPUT-Aufruf, außer dass der Parameter *Buffer* als Typ Any deklariert wird, sodass jeder beliebige Datentyp in die Warteschlange eingefügt werden kann. Dies bedeutet jedoch, dass der Parameter *Buffer* nicht daraufhin überprüft werden kann, ob er mindestens eine Größe von *BufferLength* Bytes aufweist.

C-Aufruf

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
     CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                        BUFFER, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQPUT,(HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
               BUFFER,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Eine einzelne Nachricht einreihen

Der MQPUT1-Aufruf reiht eine einzelne Nachricht in eine Warteschlange oder Verteilerliste ein oder ordnet sie einem Thema zu.

Die Warteschlange, die Verteilerliste oder das Thema muss noch nicht geöffnet sein.

Syntax

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS- Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf weggelassen und der folgende Wert für *Hconn*angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

ObjDesc
Typ: MQOD - Ein-/Ausgabe

Dies ist eine Struktur, die die Warteschlange angibt, der die Nachricht hinzugefügt wird, oder das Thema, unter dem die Nachricht veröffentlicht wird. Details siehe „MQOD - Objektdeskriptor" auf Seite 463.

Wenn die Struktur eine Warteschlange ist, muss der Benutzer autorisiert sein, die Warteschlange für die Ausgabe zu öffnen. Die Warteschlange darf **keine** Modellwarteschlange sein.

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Diese Struktur beschreibt die Attribute der gesendeten Nachricht und erhält nach ausgeführter Einreihungsanforderung Rückmeldeinformationen. Details siehe „MQMD - Nachrichtendeskriptor" auf Seite 399.

Wenn die Anwendung einen MQMD der Version 1 bereitstellt, kann den Nachrichtendaten eine MQMDE-Struktur vorangestellt werden, um Werte für die Felder anzugeben, die im MQMD der Version 2, aber nicht in Version 1 vorhanden sind. Legen Sie das Feld *Format* im MQMD auf MQFMT_MD_EXTENSION fest um anzugeben, dass eine MQMDE vorhanden ist. Weitere Informationen finden Sie unter „MQMDE – Nachrichtendeskriptorerweiterung" auf Seite 453.

Die Anwendung muss keine MQMD-Struktur bereitstellen, wenn eine gültige Nachrichtenkennung im Feld *MsgHandle* der MQGMO-Struktur oder in den Feldern *OriginalMsgHandle* oder *NewMsgHandle* der MQPMO-Struktur bereitgestellt wird. Wenn in einem dieser Felder nichts bereitgestellt wird, wird als Deskriptor der Nachricht der Deskriptor übernommen, der den Nachrichtenkennungen zugeordnet ist.

PutMsgOpts
Typ: MQPMO - Ein-/Ausgabe

Details siehe „MQPMO - Nachrichteneinreihungsoptionen" auf Seite 484.

BufferLength
Typ: MQLONG - Eingabe

Die Länge der Nachricht in *Buffer*. Null ist gültig und zeigt an, dass die Nachricht keine Anwendungsdaten enthält. Die Obergrenze hängt von verschiedenen Faktoren ab. Weitere Informationen finden Sie in der Beschreibung des Parameters *BufferLength* im MQPUT-Aufruf.

Buffer
Typ: MQBYTExBufferLength - Eingabe

Dies ist ein Puffer, der die zu sendenden Anwendungsnachrichtendaten enthält. Richten Sie den Puffer an einem Grenzwert aus, der der Spezifik der Daten in der Nachricht entspricht. 4-Byte-Ausrichtung ist für die meisten Nachrichten geeignet (einschließlich Nachrichten mit WebSphere MQ-Headerstrukturen), aber manche Nachrichten erfordern eine stringentere Ausrichtung. Zum Beispiel kann eine Nachricht, die eine binäre Ganzzahl von 64 Bit enthält, eine 8-Bit-Ausrichtung erfordern.

Wenn *Buffer* Zeichen- oder numerische Daten enthält, setzen Sie die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* auf die Werte, die den Daten entsprechen. Dies ermöglicht es dem Empfänger der Nachricht, die Daten in den verwendeten Zeichensatz und die verwendete Codierung zu konvertieren.

Anmerkung: Alle anderen Parameter beim MQPUT1-Aufruf müssen dem Zeichensatz und der Codierung des lokalen Warteschlangenmanagers entsprechen (durch das Warteschlangenmanagerattribut *CodedCharSetId* und durch MQENC_NATIVE angegeben).

In der Programmiersprache C ist der Parameter als pointer-to-void deklariert; als Parameter kann die Adresse eines jeden beliebigen Datentyps angegeben werden.

Wenn der Parameter *BufferLength* null ist, wird *Buffer* nicht referenziert. Die Adresse, die von Programmen übergeben wird, die in C oder in System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Nachrichtengruppe nicht vollständig

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Logische Nachricht nicht vollständig

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801') Nachrichtenpriorität überschreitet unterstützten Maximalwert

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838') Berichtsoptionen im Nachrichtendeskriptor nicht erkannt

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Aliasbasiswarteschlange kein gültiger Typ.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BACKED_OUT
(2003, X'7D3') Arbeitseinheit zurückgesetzt

MQRC_BUFFER_ERROR
(2004, X'7D4') Pufferparameter ungültig

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_NOT_AVAILABLE
(2345, X'929') Coupling-Facility nicht verfügbar

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') Berechtigungsprüfung für Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_ERROR
(2349, X'92D') Coupling-Facility-Struktur ungültig

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Listenüberschrift für Coupling-Facility-Struktur im Gebrauch

MQRC_CFGR_ERROR
(2416, X'970') PCF-Gruppenparameterstruktur MQCFGR in den Nachrichtendaten ungültig

MQRC_CFH_ERROR
(2235, X'8BB') PCF-Headerstruktur ungültig

MQRC_CFIF_ERROR
(2414, X'96E') PCF-Ganzzahlfilterparameterstruktur in den Nachrichtendaten ungültig

MQRC_CFIL_ERROR
(2236, X'8BC') PCF-Ganzzahllistenparameterstruktur oder PCIF*64-Ganzzahllistenparameterstruktur ungültig

MQRC_CFIN_ERROR
(2237, X'8BD') PCF-Ganzzahlparameterstruktur oder PCIF*64-Ganzzahlparameterstruktur ungültig

MQRC_CFSF_ERROR
(2415, X'96F') PCF-Zeichenfolgefilterparameterstruktur in den Nachrichtendaten ungültig

MQRC_CFSL_ERROR
(2238, X'8BE') PCF-Zeichenfolgelistenparameterstruktur ungültig

MQRC_CFST_ERROR
(2239, X'8BF') PCF-Zeichenfolgeparameterstruktur ungültig

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Exit für Clusterauslastung fehlgeschlagen.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Clusternamensauflösung fehlgeschlagen.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Clusterressourcenfehler.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') COD-Berichtsoption für XCF-Warteschlange ungültig

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Verbindung wird in Quiescemodus versetzt.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_CONTENT_ERROR
2554 (X'09FA') Nachrichteninhalt konnte nicht analysiert werden um zu ermitteln, ob die Nachricht einem Subskribenten mit erweitertem Nachrichtenselektor übergeben werden soll

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831') Referenzierte Warteschlangenkennung speichert keinen Kontext

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832') Kein Kontext für die angegebene Warteschlangenkennung vorhanden.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Parameter Datenlänge ungültig.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') DB2-Subsystem nicht verfügbar

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896') Standardübertragungswarteschlange nicht lokal.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897') Fehler bei Verwendung der Standardübertragungswarteschlange.

MQRC_DH_ERROR
(2135, X'857') Verteilungs-Headerstruktur ungültig

MQRC_DLH_ERROR
(2141, X'85D') Headerstruktur für nicht zustellbare Nachrichten ungültig

MQRC_EPH_ERROR
(2420, X'974') Eingebettete PCF-Struktur ungültig

MQRC_EXPIRY_ERROR
(2013, X'7DD') Ablaufzeit ungültig

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Rückkopplungscode ungültig

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Konflikt mit globaler Arbeitseinheit.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Gruppen-ID ungültig

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Kennung für globale Arbeitseinheit belegt.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') Keine weiteren Kennungen verfügbar.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HEADER_ERROR
(2142, X'85E') WebSphere MQ-Headerstruktur ungültig

MQRC_IIH_ERROR
(2148, X'864') Headerstruktur für IMS-Informationen ungültig

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') Globale Arbeitseinheit im Konflikt mit lokaler Arbeitseinheit.

MQRC_MD_ERROR
(2026, X'7EA') Nachrichtendeskriptor ungültig

MQRC_MDE_ERROR
(2248, X'8C8') Nachrichtendeskriptorerweiterung ungültig

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Fehlende Warteschlange für zu beantwortende Nachrichten

MQRC_MISSING_WIH
(2332, X'91C') Nachrichtendaten beginnen nicht mit MQWIH

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Nachrichtenflags ungültig

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Nachrichtenfolgenummer ungültig

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Nachrichtenlänge überschreitet den Maximalwert für die Warteschlange.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Die Nachrichtenlänge überschreitet den maximalen Wert des Warteschlangenmanagers.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Nachrichtentyp im Nachrichtendeskriptor ungültig

MQRC_MULTIPLE_REASONS
(2136, X'858') Mehrere Ursachencodes zurückgegeben.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Keine Zielwarteschlangen verfügbar

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_OBJECT_IN_USE
(2042, X'7FA') Objekt bereits mit unzulässiger Kombination von Optionen geöffnet.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X'938') Objektebene nicht kompatibel

MQRC_OBJECT_NAME_ERROR
(2152, X'868') Objektname ungültig

MQRC_OBJECT_NOT_UNIQUE
(2343, X'927') Objekt nicht eindeutig

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869') Warteschlangenmanagername für Objekt ungültig

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Objektdatensätze ungültig.

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Objekttyp ungültig.

MQRC_OD_ERROR
(2044, X'7FC') Objektdeskriptorstruktur ungültig.

MQRC_OFFSET_ERROR
(2251, X'8CB') Nachrichtensegmentoffset ungültig

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Ursprüngliche Länge ungültig

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_PAGESET_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_PCF_ERROR
(2149, X'865') PCF-Strukturen ungültig

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistenz ungültig

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800') Die Warteschlange unterstützt keine persistenten Nachrichten.

MQRC_PMO_ERROR
(2173, X'87D') Put-Message-Optionsstruktur ungültig

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Put-Message-Datensatzflags ungültig

MQRC_PRIORITY_ERROR
(2050, X'802') Nachrichtenpriorität ungültig

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') Die Veröffentlichung wurde an keinen der Subskribenten übermittelt.

MQRC_PUT_INHIBITED
(2051, X'803') Put-Aufrufe sind für die Warteschlange gesperrt.

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Nachrichteneinreihungssätze ungültig.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_FULL
(2053, X'805') Die Warteschlange enthält bereits die maximale Anzahl von Nachrichten.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_QUIESCING
(2161, X'871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') Auf dem Datenträger ist für die Warteschlange kein Speicherplatz verfügbar.

MQRC_Q_TYPE_ERROR
(2057, X'809') Warteschlangentyp ungültig.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Anzahl vorhandener Datensätze ungültig.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888') Name der fernen Warteschlange ungültig.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D) Berichtsoptionen in Nachrichtendeskriptor ungültig.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Antwortdatensätze ungültig.

MQRC_RFH_ERROR
(2334, X'91E') MQRFH- oder MQRFH2-Struktur ungültig

MQRC_RMH_ERROR
(2220, X'8AC') Headerstruktur der Referenznachricht ungültig

MQRC_SECURITY_ERROR
(2063, X'80F') Es ist ein Sicherheitsfehler aufgetreten.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') Länge der Daten im Nachrichtensegment ist null.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Ein möglicher Subskribent für die Veröffentlichung existiert, aber der Warteschlangenmanager kann nicht überprüfen, ob die Veröffentlichung an den Subskribenten gesendet werden soll

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Aufruf vom Exit für Clusterauslastung zurückgewiesen.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839') Speicherklassenfehler

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') Externes Speichermedium ist voll

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') In der aktuellen Arbeitseinheit können keine weiteren Nachrichten verarbeitet werden.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') Unterstützung für Synchronisationspunkt nicht verfügbar.

MQRC_TM_ERROR
(2265, X'8D9') Auslösenachrichtstruktur ungültig

MQRC_TMC_ERROR
(2191, X'88F') Zeichenauslösenachrichtstruktur ungültig

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822') Unbekannte Aliasbasiswarteschlange.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895') Unbekannte Standardübertragungswarteschlange.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unbekannter Objektname.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826') Unbekannter Objektwarteschlangenmanager.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827') Unbekannter ferner Warteschlangenmanager.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894') Unbekannte Übertragungswarteschlange.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Eintragung in globale Arbeitseinheit fehlgeschlagen.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') Mischung von Aufrufen für Arbeitseinheiten wird nicht unterstützt.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') Arbeitseinheit nicht zur Verwendung durch den Warteschlangenmanager verfügbar.

MQRC_WIH_ERROR
(2333, X'91D') MQWIH-Struktur ungültig

MQRC_WRONG_CF_LEVEL
(2366, X'93E') Coupling-Facility-Struktur mit falscher Version.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Falsche MQMD-Version bereitgestellt.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Übertragungswarteschlange nicht lokal.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Übertragungswarteschlange mit falscher Verwendung.

MQRC_XQH_ERROR
(2260, X'8D4') Headerstruktur der Übertragungswarteschlange ungültig

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Um Nachrichten in eine Warteschlange einzureihen, können sowohl MQPUT- als auch MQPUT1-Aufrufe verwendet werden. Welcher Aufruf jeweils zu verwenden ist, hängt von den Umständen ab.
   - Verwenden Sie den MQPUT-Aufruf, um mehrere Nachrichten in *dieselbe* Warteschlange zu platzieren.

     Zuerst wird ein MQOPEN-Aufruf mit Angabe der Option MQOO_OUTPUT ausgegeben, gefolgt von mindestens einer MQPUT-Anforderung zum Hinzufügen von Nachrichten zur Warteschlange. Zuletzt wird die Warteschlange mit einem MQCLOSE-Aufruf geschlossen. Dieses Vorgehen ermöglicht eine bessere Leistung als wiederholte MQPUT1-Aufrufe.
   - Verwenden Sie den MQPUT1-Aufruf, um nur *eine* Nachricht in die Warteschlange einzureihen.

     Dieser Aufruf bindet die MQOPEN-, MQPUT- und MQCLOSE-Aufrufe in einen einzigen Aufruf ein, wodurch die Anzahl der auszugebenden Aufrufe minimiert wird.
2. Wenn eine Anwendung eine Folge von Nachrichten in dieselbe Warteschlange einreiht, ohne Nachrichtengruppen zu verwenden,wird die Reihenfolge dieser Nachrichten beibehalten, sofern bestimmte Bedingungen erfüllt sind. Allerdings entspricht der MQPUT1-Aufruf in den meisten Umgebungen diesen Bedingungen nicht und behält so daher die Nachrichtenreihenfolge nicht bei. In diesen Umgebungen muss stattdessen der MQPUT-Aufruf verwendet werden. Weitere Informationen finden Sie im Abschnitt Hinweise zur Verwendung von MQPUT.
3. Mit dem MQPUT1-Aufruf können Nachrichten in Verteilerlisten eingereiht werden. Allgemeine Informationen dazu finden Sie in den Hinweisen zur Verwendung der MQOPEN- und MQPUT-Aufrufe.

   Verteilerlisten werden in den folgenden Umgebungen unterstützt. AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

   Die folgenden Abweichungen gelten bei der Verwendung des MQPUT1-Aufrufs:

   a. Wenn die Anwendung MQRR-Antwortdatensätze bereitstellt, müssen sie mit der MQOD-Struktur und dürfen nicht mit der MQPMO-Struktur bereitgestellt werden.

   b. Der Ursachencode MQRC_OPEN_FAILED wird niemals von MQPUT1 in den Antwortdatensätzen zurückgegeben. Wenn das Öffnen einer Warteschlange fehlschlägt, enthält der Antwortdatensatz für diese Warteschlange den Ursachencode, der aus der Operation zum Öffnen resultiert.

      Wenn eine Operation zum Öffnen einer Warteschlange mit dem Beendigungscode MQCC_WARNING erfolgreich ist, werden der Beendigungscode und der Ursachencode im Antwortdatensatz durch die Beendigungs- und Ursachencodes ersetzt, die aus der Put-Operation resultieren.

      Wie bei den MQOPEN- und MQPUT-Aufrufen setzt der Warteschlangenmanager die Antwortdatensätze (falls vorhanden) nur fest, wenn das Ergebnis des Aufrufs nicht für alle Warteschlangen in der Verteilerliste das gleiche ist. Dies wird durch den Ursachencode MQRC_MULTIPLE_REASONS beim Beenden des Aufrufs angezeigt.
4. Wenn der MQPUT1-Aufruf verwendet wird, um eine Nachricht in eine Clusterwarteschlange einzureihen, verhält sich der Aufruf so wie bei der Angabe von MQOO_BIND_NOT_FIXED beim MQOPEN-Aufruf.
5. Wenn eine Nachricht mit einer oder mehreren WebSphere MQ-Headerstrukturen zu Beginn der Anwendungsnachrichtendaten eingereiht wird, führt der Warteschlangenmanager bestimmte Überprü-

fungen der Headerstrukturen durch, um die Gültigkeit zu verifizieren. Weitere Informationen hierzu finden Sie in den Hinweisen zur Verwendung des MQPUT-Aufrufs.

6. Falls mehrere Warnungen zurückgegeben werden (siehe Parameter *CompCode*), ist der *erste* Ursachencode in der folgenden Liste der gültige Ursachencode:
   a. MQRC_MULTIPLE_REASONS
   b. MQRC_INCOMPLETE_MSG
   c. MQRC_INCOMPLETE_GROUP
   d. MQRC_PRIORITY_EXCEEDS_MAXIMUM oder MQRC_UNKNOWN_REPORT_OPTION
7. Für die Programmiersprache Visual Basic gilt Folgendes:
   - Wenn die Größe des Parameters *Buffer* kleiner als die Länge ist, die im Parameter *BufferLength* angegeben ist, schlägt der Aufruf mit Ursachencode MQRC_BUFFER_LENGTH_ERROR fehl.
   - Der Parameter *Buffer* wird als Typ `String` deklariert. Wenn es sich bei den Daten, die in die Warteschlange eingefügt werden sollen, nicht um Daten des Typs `String` handelt, verwenden Sie denMQPUT1Any-Aufruf anstelle von MQPUT1.

     Der MQPUT1Any-Aufruf hat dieselben Parameter wie der MQPUT1-Aufruf, außer dass der Parameter *Buffer* als Typ Any deklariert wird, sodass jeder beliebige Datentyp in die Warteschlange eingefügt werden kann. Dies bedeutet jedoch, dass der Parameter *Buffer* nicht daraufhin überprüft werden kann, ob er mindestens eine Größe von *BufferLength* Bytes aufweist.
8. Wenn ein MQPUT1-Aufruf mit MQPMO_SYNCPOINT ausgegeben wird, ändert sich das Standardverhalten, sodass die Put-Operation asynchron beendet wird. Dies kann eine Änderung des Verhaltens einiger Anwendungen verursachen, die bestimmte Felder in den MQOD- und MQMD-Strukturen benötigen, die zurückgegeben werden, aber jetzt undefinierte Werte enthalten. Eine Anwendung kann MQPMO_SYNC_RESPONSE angeben, um sicherzustellen, dass die Put-Operation synchron ausgeführt wird und dass alle entsprechenden Feldwerte angegeben sind.

C-Aufruf

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
       BufferLength, Buffer, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
     CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                         BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
         CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,   X
               BUFFER,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Objektattribute festlegen

Verwenden Sie den MQSET-Aufruf, um die Attribute eines durch eine Kennung angegebenen Objekts zu ändern. Das Objekt muss eine Warteschlange sein.

Syntax

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Für CICS-Anwendungen unter z/OS und Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und folgender Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Hobj
Typ: MQHOBJ - Eingabe

Diese Kennung steht für das Warteschlangenobjekt mit Attributen, die gesetzt werden sollen. Die Kennung wurde von einem früheren MQOPEN-Aufruf zurückgegeben, in dem die Option MQOO_SET angegeben war.

SelectorCount
Typ: MQLONG - Eingabe

Dies ist der Zähler für Selektoren, die im Array *Selectors* angegeben werden. Er gibt die Anzahl der Attribute an, die festgelegt werden müssen. Null ist ein gültiger Wert. Die maximal zulässige Anzahl ist 256.

Selectors
Typ: MQLONGxSelectorCount - Eingabe

Dies ist ein Array aus *SelectorCount*-Attributselektoren. Jeder Selektor gibt ein Attribut (Ganzzahl oder Zeichen) mit einem Wert an, der festgelegt werden muss.

Jeder Selektor muss für den Typ der in *Hobj* angegebenen Warteschlange gültig sein. Nur bestimmte MQIA_*- und MQCA_*-Werte sind zulässig, sie sind weiter unten aufgelistet.

Selektoren können in beliebiger Reihenfolge angegeben werden. Attributwerte für Ganzzahlattributselektoren (MQIA_*-Selektoren) müssen in *IntAttrs* in derselben Reihenfolge angegeben werden, in der diese Selektoren im Parameter *Selectors* auftreten. Attributwerte für Zeichenattributselektoren (MQCA_*-Selektoren) müssen in *CharAttrs* in derselben Reihenfolge angegeben werden, in der diese Selektoren auftreten. MQIA_*-Selektoren können mit MQCA_*-Selektoren verzahnt werden, wichtig ist allein die relative Reihenfolge innerhalb der einzelnen Typen.

Sie können denselben Selektor mehr als einmal angeben. In diesem Fall ist der letzte für einen bestimmten Selektor angegebene Wert wirksam.

Anmerkung:

1. Die Ganzzahl- und Zeichenattributselektoren werden in zwei verschiedenen Bereichen angegeben. Die MQIA_*-Selektoren befinden sich im Bereich MQIA_FIRST bis MQIA_LAST und die MQCA_*-Selektoren im Bereich MQCA_FIRST bis MQCA_LAST.

   In jedem Bereich legen die Konstanten MQIA_LAST_USED und MQCA_LAST_USED den höchsten Wert fest, der vom Warteschlangenmanager akzeptiert wird.

2. Wenn alle MQIA_*-Selektoren zuerst genannt werden, können zur Adressierung entsprechender Elemente in den Arrays *Selectors* und *IntAttrs* dieselben Elementnummern verwendet werden.
3. Wenn der Parameter *SelectorCount* auf null gesetzt ist, wird *Selectors* nicht referenziert. Die Parameteradresse, die von Programmen übergeben wird, die in C oder System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

Die Attribute, die festgelegt werden können, sind in der folgenden Tabelle aufgeführt. Mit diesem Aufruf können keine anderen Attribute festgelegt werden. Für die MQCA_*-Attributselektoren wird die Konstante, die die in *CharAttrs* erforderliche Länge der Zeichenfolge in Byte definiert, in Klammern angegeben.

Tabelle 571. MQSET-Attributselektoren für Warteschlangen

Selektor	Beschreibung	Hinweis
MQCA_TRIGGER_DATA	Auslöserdaten (MQ_TRIGGER_DATA_LENGTH)	
MQIA_DIST_LISTS	Verteilerlistenunterstützung	1
MQIA_INHIBIT_GET	Gibt an, ob GET-Operationen zulässig sind.	
MQIA_INHIBIT_PUT	Gibt an, ob PUT-Operationen zulässig sind.	
MQIA_TRIGGER_CONTROL	Auslösesteuerung.	
MQIA_TRIGGER_DEPTH	Auslöserschwelle	
MQIA_TRIGGER_MSG_PRIORITY	Nachrichtenprioritätsschwelle für Auslöser	
MQIA_TRIGGER_TYPE	Auslösertyp	

Hinweis:

1. Wird nur unter AIX, HP-UX, IBM i, Solaris, Linux, Windowssowie WebSphere MQ MQI-Clients unterstützt, die mit diesen Systemen verbunden sind.

IntAttrCount
Typ: MQLONG - Eingabe

Dies ist die Anzahl der Elemente im Array *IntAttrs*. Der Wert muss mindestens die Anzahl der MQIA_*-Selektoren im Parameter *Selectors* betragen. Null ist ein gültiger Wert, wenn keine vorhanden sind.

IntAttrs
Typ: MQLONGxIntAttrCount - Eingabe

Dies ist ein Array aus *IntAttrCount* Ganzzahlattributwerten. Diese Attributwerte müssen in derselben Reihenfolge wie die MQIA_*-Selektoren im Array *Selectors* angegeben werden.

Wenn der Parameter *IntAttrCount* oder *SelectorCount* auf null gesetzt ist, wird *IntAttrs* nicht referenziert. Die Parameteradresse, die von Programmen übergeben wird, die in C oder System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

CharAttrLength
Typ: MQLONG - Eingabe

Dies ist die Länge des Parameters *CharAttrs*in Byte. Der Wert muss mindestens der Summe der Längen der Zeichenattribute entsprechen, die im Array *Selectors* angegeben sind. Null ist ein gültiger Wert, wenn keine MQCA_*-Selektoren in *Selectors* angegeben sind.

CharAttrs
Typ: MQCHARxCharAttrLength - Eingabe

Dies ist der Puffer mit den miteinander verketteten Attributwerten. Die Länge des Puffers wird mit dem Parameter *CharAttrLength* festgelegt.

Diese Zeichenattribute müssen in derselben Reihenfolge wie die MQCA_*-Selektoren im Array *Selectors* angegeben werden. Die Länge jedes Zeichenattributs ist festgelegt(siehe *Selectors*). Wenn der für ein Attribut festgelegte Wert weniger nicht-leere Zeichen als die definierte Länge des Attributs enthält, füllen Sie den Wert in *CharAttrs* nach rechts mit Leerzeichen auf, damit der Attributwert mit der definierten Länge des Attributs übereinstimmt.

Wenn der Parameter *CharAttrLength* oder *SelectorCount* auf null gesetzt ist, wird *CharAttrs* nicht referenziert. Die Parameteradresse, die von Programmen übergeben wird, die in C oder System/390-Assembler geschrieben sind, kann in diesem Fall null sein.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CF_STRUC_FAILED
(2373, X'945') Coupling-Facility-Struktur fehlgeschlagen

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Coupling-Facility-Struktur im Gebrauch

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Listenüberschrift für Coupling-Facility-Struktur im Gebrauch

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Länge von Zeichenattributen ist ungültig

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Zeichenfolge für Zeichenattribute ist ungültig

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Warteanforderung von CICS abgelehnt

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Keine Verbindungsberechtigung

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') DB2-Subsystem nicht verfügbar

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_HOBJ_ERROR
(2019, X'7E3') Objektkennung ungültig.

MQRC_INHIBIT_VALUE_ERROR
(2020, X'7E4') Wert für Warteschlangenattribute Abrufsperre oder Einreihsperre ungültig.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Anzahl Ganzzahlattribute ungültig

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Array für Ganzzahlattribute ungültig

MQRC_NOT_OPEN_FOR_SET
(2040, X'7F8') Warteschlange nicht für Festlegen geöffnet

MQRC_OBJECT_CHANGED
(2041, X'7F9') Objektdefinition wurde nach dem Öffnen geändert.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objekt beschädigt

MQRC_PAGESET_ERROR
(2193, X'891') Fehler bei Zugriff auf Seitengruppendatensatz.

MQRC_Q_DELETED
(2052, X'804') Warteschlange wurde gelöscht.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Name des Warteschlangenmanagers ungültig oder unbekannt.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Warteschlangenmanager nicht für Verbindung verfügbar.

MQRC_Q_MGR_STOPPING
(2162, X'872') Warteschlangenmanager wird beendet

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Anzahl Selektoren ungültig

MQRC_SELECTOR_ERROR
(2067, X'813') Attributselektor ungültig

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Zähler von Selektoren zu groß.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Aufruf wurde vom Exitprogramm unterdrückt.

MQRC_TRIGGER_CONTROL_ERROR
(2075, X'81B) Wert für Auslösesteuerungsattribut ungültig

MQRC_TRIGGER_DEPTH_ERROR
(2076, X'81C) Wert für Auslöseschwellenattribut ungültig

MQRC_TRIGGER_MSG_PRIORITY_ERR
(2077, X'81D') Wert für Attribut Auslösernachrichtenpriorität ungültig.

MQRC_TRIGGER_TYPE_ERROR
(2078, X'81E') Wert für Auslöser/Typ-Priorität ungültig

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Mit diesem Aufruf kann die Anwendung eine Feldgruppe aus Ganzzahlenattributen oder eine Sammlung aus Zeichenattributfolgen oder beides angeben. Wenn keine Fehler auftreten, sind die angegebenen Attribute alle gleichzeitig gesetzt. Wenn ein Fehler auftritt (zum Beispiel wenn ein Selektor nicht gültig ist oder wenn versucht wird, ein Attribut auf einen ungültigen Wert zu setzen), schlägt der Aufruf fehl und es werden keine Attribute gesetzt.
2. Die Werte von Attributen können mit dem MQINQ-Aufruf ermittelt werden. Weitere Einzelheiten finden Sie unter „MQINQ - Objektattribute abfragen” auf Seite 700.

   Anmerkung: Nicht bei allen Attributen mit Werten, die mit dem MQINQ-Aufruf abgefragt werden können, können die Werte mit dem MQSET-Aufruf geändert werden. Zum Beispiel können keine Prozessobjekt- oder Warteschlangenattribute mit diesem Aufruf gesetzt werden.
3. Attributänderungen bleiben nach dem Neustart des Warteschlangenmanagers erhalten (anders als Änderungen an temporären dynamischen Warteschlangen, die nach einem Neustart des Warteschlangenmanagers gelöscht sind).
4. Die Attribute einer Modellwarteschlange können nicht mit dem MQSET-Aufruf geändert werden. Wenn Sie allerdings eine Modellwarteschlange mit dem MQOPEN-Aufruf mit der Option MQOO_SET öffnen, können Sie den MQSET-Aufruf verwenden, um die Attribute der dynamischen lokalen Warteschlange festzulegen, die durch den MQOPEN-Aufruf erstellt wird.
5. Wenn das Objekt, das festgelegt wird, eine Clusterwarteschlange ist, muss eine lokale Instanz der Clusterwarteschlange vorhanden sein, damit die Warteschlange erfolgreich geöffnet werden kann.

Weitere Informationen zu Objektattributen finden Sie in den folgenden Abschnitten:

- „Attribute für Warteschlangen” auf Seite 833
- „Attribute für Namenslisten” auf Seite 867
- „Attribute für Prozessdefinitionen” auf Seite 869
- „Attribute für den Warteschlangenmanager” auf Seite 797

C-Aufruf

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                   INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                   CHARATTRS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

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

Deklarieren Sie die Parameter wie folgt:

```
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
```

Aufruf in Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Nachrichteneigenschaft festlegen

Verwenden Sie den MQSET-Aufruf, um die Eigenschaft einer Nachrichtenkennung festzulegen oder zu ändern.

Syntax

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager.

Der Wert muss mit der Verbindungskennung übereinstimmen, die für die Erstellung der im Parameter *Hmsg* angegebenen Verbindungskennung verwendet wurde. Wurde die Nachrichtenkennung über MQHC_UNASSOCIATED_HCONN erstellt, muss eine gültige Verbindung in dem Thread hergestellt werden, der eine Eigenschaft für die Nachrichtenkennung festlegt, ansonsten schlägt der Aufruf mit dem Ursachencode MQRC_CONNECTION_BROKEN fehl.

Hmsg
Typ: MQHMSG - Eingabe

Dies ist die zu ändernde Nachrichtenkennung. Der Wert wurde durch einen früheren MQCRTMH-Aufruf zurückgemeldet.

SetPropOpts
Typ: MQSMPO - Eingabe

Steuert, wie Nachrichteneigenschaften festgelegt werden.

Über diese Struktur können Anwendungen Optionen für das Festlegen von Nachrichteneigenschaften definieren. Bei der Struktur handelt es sich um einen Eingabeparameter im MQSETMP-Aufruf. Weitere Informationen hierzu finden Sie unter MQSMPO.

Name
Typ: MQCHARV- Eingabe

Dies ist der Name der festzulegenden Eigenschaft.

Weitere Informationen zur Verwendung von Eigenschaftsnamen finden Sie in den Abschnitten Eigenschaftsnamen und Einschränkungen bei Eigenschaftsnamen.

PropDesc
Typ: MQPD - Ein-/Ausgabe

Mit dieser Struktur werden die Attribute einer Eigenschaft beschrieben, wie:

- was geschieht, wenn die Eigenschaft nicht unterstützt wird
- zu welchem Nachrichtenkontext die Eigenschaft gehört
- in welche Nachrichten die Eigenschaft bei der Verarbeitung kopiert wird

Weitere Informationen über diese Struktur finden Sie unter MQPD.

Type
Typ: MQLONG - Eingabe

Der Datentyp der festzulegenden Eigenschaft. Folgende sind möglich:

MQTYPE_BOOLEAN
Ein boolescher Wert. *ValueLength* muss 4 sein.

MQTYPE_BYTE_STRING
Eine Bytefolge. *ValueLength* muss null oder größer sein.

MQTYPE_INT8
Eine 8 Bit lange ganze Zahl mit Vorzeichen. *ValueLength* muss 1 sein.

MQTYPE_INT16
Eine 16 Bit lange ganze Zahl mit Vorzeichen. *ValueLength* muss 2 sein.

MQTYPE_INT32
Eine 32 Bit lange ganze Zahl mit Vorzeichen. *ValueLength* muss 4 sein.

MQTYPE_INT64
Eine 64 Bit lange ganze Zahl mit Vorzeichen. *ValueLength* muss 8 sein.

MQTYPE_FLOAT32
Eine 32 Bit lange Gleitkommazahl. *ValueLength* muss 4 sein.

Hinweis: Dieser Typ wird in Anwendungen, die IBM COBOL for z/OS verwenden, nicht unterstützt.

MQTYPE_FLOAT64
Eine 64 Bit lange Gleitkommazahl. *ValueLength* muss 8 sein.

Hinweis: Dieser Typ wird in Anwendungen, die IBM COBOL for z/OS verwenden, nicht unterstützt.

MQTYPE_STRING
Eine Zeichenfolge. *ValueLength* muss null oder höher oder der spezielle Wert MQVL_NULL_TERMINATED sein.

MQTYPE_NULL
Die Eigenschaft existiert, hat aber einen Nullwert. *ValueLength* muss null sein.

ValueLength
Typ: MQLONG - Eingabe

Länge des Eigenschaftswerts im Parameter *Value* in Byte. Null ist nur für Nullwerte oder für Zeichenfolgen oder Bytefolgen gültig. Null weist darauf hin, dass die Eigenschaft existiert, der Wert aber keine Zeichen oder Bytes enthält.

Der Wert muss größer-gleich null oder der folgende spezielle Wert sein, wenn für den Parameter *Type* MQTYPE_STRING gesetzt ist:

MQVL_NULL_TERMINATED
Der Wert wird durch die erste in der Zeichenfolge vorkommende Null begrenzt. Die Null wird nicht als Teil der Zeichenfolge eingeschlossen. Dieser Wert ist ungültig, wenn nicht außerdem MQTYPE_STRING gesetzt ist.

Hinweis: Das zur Begrenzung einer Zeichenfolge verwendete Nullzeichen, wenn MQVL_NULL_TERMINATED gesetzt ist, ist eine Null aus dem Zeichensatz des Wertes.

Wert
Typ: MQBYTExValueLength - Eingabe

Der Wert der festzulegenden Eigenschaft. Der Puffer muss auf einen auf die Art der im Wert enthaltenen Daten abgestimmten Grenzwert ausgerichtet sein.

In der Programmiersprache C ist der Parameter als pointer-to-void deklariert; als Parameter kann die Adresse eines jeden beliebigen Datentyps angegeben werden.

Ist *ValueLength* null, gibt es keinen Verweis auf *Value*. In diesem Fall kann die von in C oder System/390 Assembler geschriebenen Programmen übergebene Parameteradresse null sein.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_RFH_FORMAT_ERROR
(2421, X'0975') Ein MQRFH2-Ordner, der Eigenschaften enthält, konnte nicht analysiert werden.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adapter nicht verfügbar.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Adapterservicemodul kann nicht geladen werden.

MQRC_ASID_MISMATCH
(2157, X'86D') Unterschiedliche IDs für Primär- und Ausgangsadressraum.

MQRC_BUFFER_ERROR
(2004, X'07D4') Wertparameter ungültig.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Längenparameter des Werts nicht gültig.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_HMSG_ERROR
(2460, X'099C') Nachrichtenkennungsverweis ungültig.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') Nachrichtenkennung wird bereits verwendet.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Optionen ungültig oder nicht konsistent.

MQRC_PD_ERROR
(2482, X'09B2') Struktur des Eigenschaftsdeskriptors nicht gültig.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Ungültiger Eigenschaftsname.

MQRC_PROPERTY_TYPE_ERROR
(2473, X'09A9') Ungültiger Eigenschaftsdatentyp.

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Zahlenformatfehler in Wertdaten gefunden.

MQRC_SMPO_ERROR
(2463, X'099F') Struktur zur Festlegung der Nachrichteneigenschaften ungültig.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') ID des codierten Zeichensatzes des Eigenschaftsnamens nicht gültig.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C-Aufruf

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
     CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDESC, TYPE,
                          VALUELENGTH, VALUE, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
              Value, CompCode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
        CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
               VALUE,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Statusinformationen abrufen

Verwenden Sie den MQSTAT-Aufruf, um Statusinformationen abzurufen. Die Art der zurückgegebenen Statusinformationen wird durch den im Aufruf angegebenen Wert für den Parameter "Type" festgelegt.

Syntax

MQSTAT (*Hconn*, *Type*, *Stat*, *Compcode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS- Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf weggelassen und der folgende Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Type
Typ: MQLONG - Eingabe

Art der angeforderten Statusinformationen. Gültige Werte sind:

MQSTAT_TYPE_ASYNC_ERROR
Gibt Informationen zu vorherigen asynchronen Put-Operationen zurück.

MQSTAT_TYPE_RECONNECTION

Gibt Informationen zur Verbindungswiederherstellung zurück. Wenn die Verbindung wiederhergestellt wird oder nicht wiederhergestellt werden konnte, beschreiben die Informationen den Fehler, der dazu geführt hat, dass die Verbindung mit der Wiederherstellung begonnen hat.

Dieser Wert ist nur für Clientverbindungen gültig. Für andere Verbindungsarten schlägt der Aufruf mit dem Ursachencode **MQRC_ENVIRONMENT_ERROR** fehl.

MQSTAT_TYPE_RECONNECTION_ERROR

Gibt Informationen zu einem vorherigen Fehler im Zusammenhang mit dem Wiederherstellen der Verbindung zurück. Wenn die Verbindung nicht wiederhergestellt werden konnte, beschreiben die Informationen den Fehler, der dazu geführt hat, dass die Wiederherstellung der Verbindung fehlgeschlagen ist.

Dieser Wert ist nur für Clientverbindungen gültig. Für andere Verbindungsarten schlägt der Aufruf mit dem Ursachencode **MQRC_ENVIRONMENT_ERROR** fehl.

Stat
Typ: MQSTS - Ein-/Ausgabe

Struktur der Statusinformationen. Details siehe Abschnitt „MQSTS – Statusberichtsstruktur” auf Seite 580.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der *CompCode*qualifiziert.

Wenn *CompCode* auf MQCC_OK gesetzt ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_API_EXIT_ERROR
(2374, X'946') API-Exit fehlgeschlagen.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API-Exit kann nicht geladen werden.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') MQI-Aufruf wurde vor Beendigung des vorherigen Aufrufs eingegeben.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Verbindung mit Warteschlangenmanager verloren

MQRC_CONNECTION_STOPPING
(2203, X'89B') Verbindung wird beendet.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') Die angeforderte Funktion ist in der aktuellen Umgebung nicht verfügbar.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_Q_MGR_STOPPING
(2162,X'872' - Warteschlangenmanager wird angehalten

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_STAT_TYPE_ERROR
(2430, X'97E' Fehler mit MQSTAT-Typ

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_STS_ERROR
(2426, X'97A') Fehler mit MQSTS-Struktur

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Ein Aufruf von MQSTAT mit dem Typ MQSTAT_TYPE_ASYNC_ERROR gibt Informationen zu vorherigen asynchronen MQPUT -und MQPUT1 -Operationen zurück. Die MQSTS-Struktur, die bei der Rückgabe vom Aufruf MQSTAT übergeben wird, enthält die ersten aufgezeichneten asynchronen Warnungs- oder Fehlerinformationen für diese Verbindung. Treten danach weitere Fehler oder Warnungen auf, werden diese Werte durch sie normalerweise nicht geändert. Wenn jedoch ein Fehler mit dem Beendigungscode MQCC_WARNING auftritt, wird stattdessen ein nachfolgender Fehler mit dem Beendigungscode MQCC_FAILED zurückgegeben.
2. Wenn seit dem Herstellen der Verbindung oder seit dem letzten Aufruf von MQSTAT keine Fehler aufgetreten sind, wird der Wert CompCode von MQCC_OK und die Ursache MQRC_NONE in der Struktur MQSTS zurückgegeben.
3. Die Anzahl der asynchronen Aufrufe, die unter der Verbindungskennung verarbeitet wurden, wird über drei Zählerfelder zurückgegeben: PutSuccessCount, PutWarningCount und PutFailure-Count. Diese Zähler werden vom Warteschlangenmanager jedes Mal erhöht, wenn eine asynchrone Operation erfolgreich verarbeitet wird, eine Warnung enthält oder fehlschlägt (beachten Sie, dass zu Abrechnungszwecken eine Put-Operation für eine Verteilerliste einmal pro Zielwarteschlange gezählt wird und nicht einmal pro Verteilerliste). Ein Zähler wird nicht über den maximalen positiven Wert AMQ_LONG_MAX hinaus erhöht.
4. Ein erfolgreicher Aufruf an MQSTAT führt dazu, dass alle vorherigen Fehlerinformationen oder -zähler zurückgesetzt werden.
5. Das Verhalten von MQSTAT hängt vom Wert des Parameters MQSTAT Type ab, den Sie bereitstellen.
6. **MQSTAT_TYPE_ASYNC_ERROR**
    a. Ein Aufruf von MQSTAT mit dem Typ MQSTAT_TYPE_ASYNC_ERROR gibt Informationen zu vorherigen asynchronen MQPUT -und MQPUT1 -Operationen zurück. Die MQSTS-Struktur, die bei der Rückgabe vom Aufruf MQSTAT übergeben wird, enthält die ersten aufgezeichneten asynchronen Warnungs- oder Fehlerinformationen für diese Verbindung. Treten danach weitere Fehler oder Warnungen auf, werden diese Werte durch sie normalerweise nicht geändert. Wenn jedoch ein Fehler mit dem Beendigungscode MQCC_WARNING auftritt, wird stattdessen ein nachfolgender Fehler mit dem Beendigungscode MQCC_FAILED zurückgegeben.
    b. Wenn seit dem Herstellen der Verbindung oder seit dem letzten Aufruf von MQSTAT keine Fehler aufgetreten sind, wird der Wert CompCode von MQCC_OK und die Ursache MQRC_NONE in der Struktur MQSTS zurückgegeben.

c. Die Anzahl der asynchronen Aufrufe, die unter der Verbindungskennung verarbeitet wurden, wird über drei Zählerfelder zurückgegeben: PutSuccessCount, PutWarningCount und PutFailureCount. Diese Zähler werden vom Warteschlangenmanager jedes Mal erhöht, wenn eine asynchrone Operation erfolgreich verarbeitet wird, eine Warnung enthält oder fehlschlägt (beachten Sie, dass zu Abrechnungszwecken eine Put-Operation für eine Verteilerliste einmal pro Zielwarteschlange gezählt wird und nicht einmal pro Verteilerliste). Ein Zähler wird nicht über den maximalen positiven Wert AMQ_LONG_MAX hinaus erhöht.

d. Ein erfolgreicher Aufruf an MQSTAT führt dazu, dass alle vorherigen Fehlerinformationen oder -zähler zurückgesetzt werden.

MQSTAT_TYPE_RECONNECTION

Angenommen, Sie rufen MQSTAT innerhalb eines Ereignishandlers während der Wiederherstellung auf, während Type auf MQSTAT_TYPE_RECONNECTION festgelegt ist. Sehen Sie sich diese Beispiele an.

Der Client versucht, die Verbindung wiederherzustellen oder die Verbindungswiederherstellung ist fehlgeschlagen.

CompCode in der MQSTS-Struktur ist MQCC_FAILED und Reason kann MQRC_CONNECTION_BROKEN oder MQRC_Q_MGR_QUIESCING sein. ObjectType ist MQOT_Q_MGR, ObjectName ist der Name des Warteschlangenmanagers und ObjectQMgrName ist leer.

Der Client hat die Verbindungswiederherstellung erfolgreich abgeschlossen oder die Verbindung war nie getrennt.

CompCode in der MQSTS -Struktur ist MQCC_OK und Reason ist MQRC_NONE

Nachfolgende Aufrufe an MQSTAT geben dieselben Ergebnisse zurück.

MQSTAT_TYPE_RECONNECTION_ERROR

Angenommen, Sie rufen MQSTAT als Reaktion auf den Empfang von MQRC_RECONNECT_FAILED nach einem MQI-Aufruf auf, während Type auf MQSTAT_TYPE_RECONNECTION_ERROR gesetzt ist. Sehen Sie sich diese Beispiele an.

Als eine Warteschlange während der Verbindungswiederherstellung zu einem anderen Warteschlangenmanager erneut geöffnet wurde, ist ein Autorisierungsfehler aufgetreten.

CompCode in der MQSTS-Struktur ist MQCC_FAILED und Reason ist der Grund für das Fehlschlagen der Verbindungswiederherstellung, wie z. B. MQRC_NOT_AUTHORIZED. ObjectType ist der Typ des Objekts, das das Problem verursacht hat, wie z. B. MQOT_QUEUE, ObjectName ist der Name der Warteschlange und ObjectQMgrName ist der Name des Warteschlangenmanagers, dem die Warteschlange gehört.

Während der Verbindungswiederherstellung ist ein Socketverbindungsfehler aufgetreten.

CompCode in der MQSTS-Struktur ist MQCC_FAILED und Reason ist der Grund für das Fehlschlagen der Verbindungswiederherstellung, wie z. B. MQRC_HOST_NOT_AVAILABLE. ObjectType ist MQOT_Q_MGR, ObjectName ist der Name des Warteschlangenmanagers und ObjectQMgrName ist leer.

Nachfolgende Aufrufe an MQSTAT geben dieselben Ergebnisse zurück.

C-Aufruf

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN Hconn;        /* Connection Handle */
MQLONG StatType;    /* Status type */
MQSTS Stat;           /* Status information structure */
MQLONG CompCode;    /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Aufruf in COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**    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.
```

Aufruf in PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl StatType    fixed bin(31); /* Status type */
dcl Stat            like MQSTS;   /* Status information structure */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason        fixed bin(31); /* Reason code qualifying CompCode */
```

System/390 Assembleraufruf

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 – Subskription registrieren

Verwenden Sie den Aufruf MQSUB, um die Anwendungssubskription für ein bestimmtes Thema zu registrieren.

Syntax

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Reason*)

Parameter

Hconn

Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Unter z/OS für CICS- Anwendungen und unter IBM i für Anwendungen, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf weggelassen und der folgende Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

SubDesc
Typ: MQSD - Ein-/Ausgabe

Dies ist eine Struktur, die das verwendete Objekt angibt, das von der Anwendung registriert wird. Weitere Informationen finden Sie im Abschnitt „MQSD - Subskriptionsdeskriptor" auf Seite 552.

Hobj
Typ: MQHOBJ - Ein-/Ausgabe

Diese Kennung steht für den Zugriff, der für den Abruf der Nachrichten, die an diese Subskription gesendet werden, eingerichtet wurde. Diese Nachrichten können entweder in einer bestimmten Warteschlange gespeichert werden oder der Warteschlangenmanager verwaltet die Speicherung der Nachrichten, ohne dafür eine bestimmte Warteschlange zu verwenden.

Um eine bestimmte Warteschlange verwenden zu können, müssen Sie diese der Subskription zuordnen, wenn die Subskription eingerichtet wird. Dazu haben Sie zwei Möglichkeiten:

- Sie verwenden den MQSC-Befehl DEFINE SUB und geben in dem Befehl den Namen eines Warteschlangenobjekts an.
- Sie übergeben die Kennung beim Aufruf von MQSUB mit der Option MQSO_CREATE.

  Wenn die Kennung als Eingabeparameter an den Aufruf übergeben wird, muss es sich um eine gültige Objektkennung handeln, die von einem vorherigen MQOPEN-Aufruf einer Warteschlange zurückgegeben wurde, in dem mindestens eine der folgenden Optionen angegeben war:

  – MQOO_INPUT_*
  – MQOO_BROWSE
  – MQOO_OUTPUT (wenn es sich um eine ferne Warteschlange handelt)

  Wenn dies nicht der Fall ist, schlägt der Aufruf mit MQRC_HOBJ_ERROR fehl. Es kann keine Objektkennung für eine Aliaswarteschlange sein, die in ein Themenobjekt aufgelöst wird. Wenn doch, schlägt der Aufruf mit MQRC_HOBJ_ERROR fehl.

Wenn der Warteschlangenmanager die Speicherung von Nachrichten, die an diese Subskription gesendet werden, verwalten soll, muss dies bei der Einrichtung der Subskription mit der Option MQSO_MANAGED festgelegt werden. Der Warteschlangenmanager gibt die Kennung dann als Ausgabeparameter im Aufruf zurück. Die zurückgegebene Kennung wird als verwaltete Kennung bezeichnet. Wenn MQHO_NONE angegeben wird, aber nicht MQSO_MANAGED, schlägt der Aufruf mit MQRC_HOBJ_ERROR fehl.

Wenn der Warteschlangenmanager eine verwaltete Kennung an Sie zurückgibt, können Sie diese in einem MQGET- oder MQCB-Aufruf mit oder ohne Suchoptionen, in einem MQINQ-Aufruf oder in einem MQCLOSE-Aufruf verwenden. Die Kennung kann nicht in einem MQPUT-, MQSUB- oder MQSET-Aufruf verwendet werden; bei einem entsprechenden Versuch schlägt der Aufruf mit MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR oder MQRC_NOT_OPEN_FOR_SET fehl.

Wenn diese Subskription mithilfe der Option MQSO_RESUME in der MQSD-Struktur wiederaufgenommen wird, kann die Kennung in diesem Parameter an die Anwendung zurückgegeben werden, indem MQSO_MANAGED auf MQHO_NONE gesetzt wird. Dies kann unabhängig davon erfolgen, ob die Subskription eine verwaltete Kennung verwendet oder nicht, und es kann hilfreich sein, um Subskriptionen bereitzustellen, die mit dem Befehl DEFINE SUB mit der Kennung für die Subskriptionswarteschlange, die in diesem Befehl definiert ist, erstellt wurden. Falls eine zu Verwaltungszwecken erstellte Subskription wiederaufgenommen wird, wird die Warteschlange mit MQOO_INPUT_AS_Q_DEF und MQOO_BROWSE geöffnet. Wenn Sie andere Optionen angeben müssen, muss die Anwendung die Subskriptionswarteschlange explizit öffnen und die Objektkennung im Aufruf angeben. Tritt beim Öffnen der Warteschlange ein Fehler auf, schlägt der Aufruf mit MQRC_INVALID_DESTINATION fehl. Wenn *Hobj* bereitgestellt wird, muss es äquivalent zu *Hobj* im ursprünglichen MQSUB-Aufruf sein. Dies bedeutet, dass es sich bei der Bereitstellung einer Objektkennung, die von einem MQOPEN-Aufruf zurückgegeben wurden, um die Kennung für dieselbe Warteschlange handeln muss, die zuvor

verwendet wurde. Ist es nicht dieselbe Warteschlange, schlägt der Aufruf mit MQRC_HOBJ_ERROR fehl.

Wenn diese Subskription mit der Option MQSO_ALTER in der MQSD-Struktur geändert wird, kann eine andere *Hobj* bereitgestellt werden. Alle Veröffentlichungen, die an die Warteschlange zugestellt wurden und die zuvor über diesen Parameter identifiziert wurden, verbleiben in dieser Warteschlange und es liegt in der Verantwortung der Anwendung, diese Nachrichten abzurufen, wenn der Parameter *Hobj* jetzt eine andere Warteschlange darstellt.

Die Tabelle zeigt die Verwendung dieses Parameters mit verschiedenen Subskriptionsoptionen:

Optionen	***Hobj***	**Beschreibung**
MQSO_CREATE + MQSO_MANAGED	Wird bei Eingabe ignoriert	Erstellt eine Subskription, bei der die Speicherung von Nachrichten durch den Warteschlangenmanager verwaltet wird.
MQSO_CREATE	Gültige Objektkennung	Erstellt eine Subskription, bei der eine bestimmte Warteschlange als Ziel für Nachrichten angegeben wird.
MQSO_RESUME	MQHO_NONE	Nimmt eine zuvor erstellte Subskription wieder auf, egal ob sie verwaltet wurde oder nicht, und der Warteschlangenmanager gibt die Objektkennung zur Verwendung durch die Anwendung zurück.
MQSO_RESUME	Gültige, übereinstimmende Objektkennung	Nimmt eine zuvor erstellte Subskription wieder auf, die eine bestimmte Warteschlange als Ziel für Nachrichten nutzte, und verwendet eine Objektkennung mit bestimmten Optionen zum Öffnen.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Ändert eine vorhandene Subskription, die zuvor eine bestimmte Warteschlange verwendete, sodass es sich jetzt um eine verwaltete Subskription handelt. Die Zielklasse (verwaltet oder nicht) kann nicht geändert werden.
MQSO_ALTER	Gültige Objektkennung	Ändert eine vorhandene Subskription, egal ob sie verwaltet wurde oder nicht, sodass sie jetzt eine bestimmte Warteschlange verwendet. Wenn die Option MQSO_MANAGED nicht verwendet wird, kann die angegebene Warteschlange geändert werden, aber nicht die Zielklasse (verwaltet oder nicht).

Unabhängig davon, ob sie bereitgestellt oder zurückgegeben wurde, muss *Hobj* in nachfolgenden MQGET-oder MQCB-Aufrufen angegeben werden, die die an diese Subskription gesendeten Veröffentlichungsnachrichten empfangen möchten.

Die Kennung *Hobj* ist nicht mehr gültig, wenn der MQCLOSE-Aufruf ausgegeben wird oder wenn die Verarbeitungseinheit, die den Geltungsbereich der Kennung definiert, beendet wird (bis die Verbindung zur Anwendung getrennt wird). Der Geltungsbereich für die zurückgegebene Kennung ist

derselbe wie der für die Verbindungskennung, die im Aufruf angegeben wird. Informationen zum Geltungsbereich der Kennung finden Sie im Abschnitt Hconn (MQHCONN) - Ausgabe. Ein MQCLOSE der Kennung *Hobj* hat keine Auswirkung auf die Kennung *Hsub* .

Hsub
Typ: MQHOBJ - Ausgabe

Diese Kennung steht für die Subskription, die eingerichtet wurde. Sie kann für zwei weitere Operationen verwendet werden:

- Sie kann in einem nachfolgenden MQSUBRQ-Aufruf verwendet werden, um die Veröffentlichungen abzurufen, die gesendet werden, wenn bei der Einrichtung der Subskription die Option MQSO_PUBLICATIONS_ON_REQUEST angegeben wurde.
- Sie kann in einem nachfolgenden MQCLOSE-Aufruf verwendet werden, um die eingerichtete Subskription zu entfernen. Die Kennung *Hsub* wird ungültig, wenn der MQCLOSE-Aufruf ausgegeben wird oder wenn die Verarbeitungseinheit, die den Geltungsbereich definiert, beendet wird. Der Geltungsbereich für die zurückgegebene Kennung ist derselbe wie der für die Verbindungskennung, die im Aufruf angegeben wird. Ein MQCLOSE der Kennung *Hsub* hat keine Auswirkung auf die Kennung *Hobj* .

Diese Kennung kann nicht an einen MQGET- oder MQCB-Aufruf übergeben werden. Sie müssen den Parameter *Hobj* verwenden. Die Kennung kann nur in den WebSphere MQ-Aufrufen MQCLOSE und MQSUBRQ verwendet werden. Wird sie an einen anderen WebSphere MQ-Aufruf übergeben, führt dies zu dem Fehler MQRC_HOBJ_ERROR.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Ausführung.

MQCC_WARNING
Warnung (teilweise Ausführung).

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der *CompCode*qualifiziert.

Wenn *CompCode* MQCC_OK ist, lautet der Ursachencode wie folgt:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* MQCC_FAILED ist, ist der Ursachencode einer der folgenden:

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Clusternamensauflösung fehlgeschlagen.

MQRC_DURABILITY_NOT_ALLOWED
2436 (X'0984') Ein MQSUB-Aufruf mit der Option MQSO_DURABLE ist fehlgeschlagen.

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') Die angeforderte Funktion ist in der aktuellen Umgebung nicht verfügbar.

MQRC_HOBJ_ERROR
2019 (X'07E3') Objektkennung Hobj ungültig.

MQRC_IDENTITY_MISMATCH
2434 (X'0982') Subskriptionsname entspricht dem einer vorhandenen Subskription.

MQRC_NOT_AUTHORIZED
2035 (X'07F3') Der Benutzer ist nicht zur Ausführung der Operation berechtigt.

MQRC_OBJECT_STRING_ERROR
2441 (X'0989') Objektzeichenfolgefeld ungültig.

MQRC_OPTIONS_ERROR
2046 (X'07FE') Optionsparameter oder Feld enthält ungültige Optionen oder eine ungültige Kombination von Optionen.

MQRC_Q_MGR_QUIESCING
2161, (X'0871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_RECONNECT_Q_MGR_REQD
2555 (X'09FB'X) Option MQCNO_RECONNECT_Q_MGR ist erforderlich.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Ständige Veröffentlichungen, die für die subskribierte Themenzeichenfolge vorhanden sind, können nicht abgerufen werden.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Die ständigen Veröffentlichungen, die für die subskribierte Themenzeichenfolge vorhanden sind, können nicht an die Zielwarteschlange der Subskription und nicht an die Warteschlange für nicht zustellbare Nachrichten übermittelt werden.

MQRC_SD_ERROR
2424 (X'0978') Subskriptionsdeskriptor (MQSD) ungültig.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Die Auswahlzeichenfolge entspricht nicht der WebSphere MQ-Selektorsyntax und es war kein erweiterter Nachrichtenauswahlanbieter verfügbar.

MQRC_SELECTION_STRING_ERROR
2519 (X'09D7') Die Auswahlzeichenfolge muss so angegeben werden, wie in der Dokumentation der MQCHARV-Struktur beschrieben.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Es wurde ein MQOPEN-, MQPUT1- oder MQSUB-Aufruf ausgegeben, aber eine Auswahlzeichenfolge angegeben, die einen Syntaxfehler enthielt.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') Feld SubUserData ist ungültig.

MQRC_SUB_NAME_ERROR
2440 (X'0988') Feld SubName ist ungültig.

MQRC_SUB_ALREADY_EXISTS
2432 (X'0980') Subskription bereits vorhanden.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') Feld SubUserData ist ungültig.

MQRC_TOPIC_STRING_ERROR
2425 (X'0979') Themenzeichenfolge ungültig.

MQRC_UNKNOWN_OBJECT_NAME
2085 (X'0825') Angegebenes Objekt kann nicht gefunden werden.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

1. Die Subskription wird für ein Thema eingerichtet, das entweder durch den Kurznamen eines vordefinierten Themenobjekts oder den vollständigen Namen der Themenzeichenfolge bezeichnet wird oder durch die Verkettung von zwei Teilen gebildet wird. Siehe die Beschreibung von *ObjectName* und *ObjectString* in „MQSD - Subskriptionsdeskriptor” auf Seite 552.
2. Wenn ein MQSUB-Aufruf ausgegeben wird, führt der Warteschlangenmanager Sicherheitsprüfungen durch, um festzustellen, ob die Benutzer-ID, unter der die Anwendung ausgeführt wird, über die richtige Berechtigungsstufe verfügt; erst danach wird der Zugriff erlaubt. Das entsprechende Themenobjekt befindet sich in der Themenhierarchie und es findet eine Berechtigungsprüfung für das Themenobjekt statt, um sicherzustellen, dass eine Subskriptionsberechtigung besteht. Wenn die Option MQSO_MA-

NAGED verwendet wird, findet eine Berechtigungsprüfung für die Zielwarteschlange statt, um sicherzustellen, dass eine Ausgabeberechtigung besteht. Wenn die Option MQSO_MANAGED verwendet wird, findet keine Berechtigungsprüfung für die verwaltete Warteschlange in Bezug auf Ausgabe- oder Abfragezugriff statt.

3. Wenn Sie keine Hobj-Kennung als Eingabe bereitstellen, werden durch den MQSUB-Aufruf zwei Kennungen zugeordnet: Eine Objektkennung (Hobj) und eine Subskriptionskennung (Hsub).
4. Die Hobj-Kennung, die bei Angabe der Option MQSO_MANAGED im Aufruf MQSUB zurückgegeben wird, kann abgefragt werden, um Attribute wie den Rücksetzschwellenwert und den Namen der Warteschlange zum Wiedereinreihen überzähliger zurückgesetzter Nachrichten zu finden. Sie können auch den Namen der verwalteten Warteschlange abfragen, dürfen aber nicht versuchen, diese Warteschlange direkt zu öffnen.
5. Subskriptionen können zu Gruppen zusammengefasst werden, sodass auch dann nur eine einzelne Veröffentlichung an die Subskriptionsgruppe übermittelt wird, wenn mehrere Mitglieder der Gruppe die Veröffentlichung subskribiert haben. Die Gruppierung von Subskriptionen erfolgt mit der Option MQSO_GROUP_SUB. Damit Subskriptionen gruppiert werden können, müssen sie folgende Voraussetzungen erfüllen:

   - Sie müssen dieselbe benannte Warteschlange (die nicht die Option MQSO_MANAGED verwendet) desselben Warteschlangenmanagers verwenden – angegeben durch den Parameter Hobj im MQSUB-Aufruf.
   - Sie müssen denselben Wert für das Attribut SubCorrelId verwenden.
   - Sie müssen denselben Wert für das Attribut SubLevel verwenden.

   Diese Attribute definieren die Subskriptionen, die als Mitglieder der Gruppe betrachtet werden, und können auch nicht geändert werden, wenn eine Subskription gruppiert ist. Wird das Attribut SubLevel geändert, führt dies zu dem Fehler MQRC_SUBLEVEL_NOT_ALTERABLE, und wird eines der anderen Attribute geändert (die geändert werden können, wenn eine Subskription nicht gruppiert ist) führt dies zu dem Fehler MQRC_GROUPING_NOT_ALTERABLE.
6. Felder in der MQSD-Struktur werden bei der Rückgabe eines MQSUB-Aufrufs, in dem die Option MQSO_RESUME verwendet wird, ausgefüllt. Der zurückgegebene MQSD kann direkt an einen MQSUB-Aufruf übergeben werden, der die Option MQSO_ALTER mit Änderungen verwendet, die Sie für die Subskription vornehmen müssen, die für den MQSD gilt. Bei einigen Feldern sind besondere Hinweise zu beachten (siehe Tabelle).

MQSD-Ausgabe von MQSUB	
Feldname in MQSD	**Besondere Hinweise**
Zugriffs- oder Erstellungsoptionen	Einige dieser Optionen können bei der Rückgabe vom Aufruf MQSUB zurückgesetzt werden. Wenn Sie den MQSD anschließend erneut in einem MQSUB-Aufruf verwenden, muss die erforderliche Option explizit gesetzt werden.
Optionen für Lebensdauer, Ziel, Registrierung und Platzhalter	Diese Optionen sind auf geeignete Weise gesetzt.
Veröffentlichungsoptionen	Diese Optionen sind auf geeignete Weise gesetzt, außer MQSO_NEW_PUBLICATIONS_ONLY, für die nur MQSO_CREATE gültig ist.
Sonstige Optionen	Diese Optionen sind bei der Rückgabe eines MQSUB-Aufrufs unverändert. Sie steuern, wie der API-Aufruf ausgegeben wird, und werden nicht mit der Subskription gespeichert. Sie müssen so gesetzt werden, wie es für einen nachfolgenden MQSUB-Aufruf, der den MQSD wiederverwendet, erforderlich ist.

MQSD-Ausgabe von MQSUB *(Forts.)*	
Feldname in MQSD	**Besondere Hinweise**
ObjectName	Dieses Nur-Eingabe-Feld ist bei der Rückgabe eines MQSUB-Aufrufs unverändert.
ObjectString	Dieses Nur-Eingabe-Feld ist bei der Rückgabe eines MQSUB-Aufrufs unverändert. Der verwendete vollständige Themenname wird im Feld *ResObjectString* zurückgegeben, wenn ein Puffer bereitgestellt wird.
AlternateUserId und AlternateSecurityId	Diese Nur-Eingabe-Felder sind bei der Rückgabe eines MQSUB-Aufrufs unverändert. Sie steuern, wie der API-Aufruf ausgegeben wird, und werden nicht mit der Subskription gespeichert. Sie müssen so gesetzt werden, wie es für einen nachfolgenden MQSUB-Aufruf, der den MQSD wiederverwendet, erforderlich ist.
SubExpiry	Bei der Rückgabe eines MQSUB-Aufrufs mit der Option MQSO_RESUME ist dieses Feld auf die ursprüngliche Ablaufzeit der Subskription gesetzt, nicht auf die verbleibende Ablaufzeit. Wenn Sie den MQSD anschließend in einem MQSUB-Aufruf mit der Option MQSO_ALTER verwenden, setzen Sie die Ablaufzeit der Subskription zurück, sodass der Countdown erneut beginnt.
SubName	Dieses Feld ist ein Eingabefeld in einem MQSUB-Aufruf, das bei der Ausgabe nicht geändert wird.
SubUserData und SelectionString	Diese Felder variabler Länge werden bei der Ausgabe von einem MQSUB-Aufruf mit der Option MQSO_RESUME zurückgegeben, wenn ein Puffer bereitgestellt wird, sowie eine positive Puffergröße in *VSBufSize*. Wenn kein Puffer bereitgestellt wird, wird nur die Länge im Feld *VSLength* der MQCHARV-Struktur zurückgegeben. Wenn der bereitgestellte Puffer kleiner als der für die Rückgabe des Felds erforderliche Speicherplatz ist, werden nur *VSBufSize* Byte im bereitgestellten Puffer zurückgegeben. Wenn Sie dann den MQSD in einem MQSUB-Aufruf mit der MQSO_ALTER-Option wiederverwenden und kein Puffer bereitgestellt wird, aber ein *VSLength* ungleich null angegeben wird, wird das Feld nicht geändert, wenn diese Länge mit der vorhandenen Länge des Felds übereinstimmt.

MQSD-Ausgabe von MQSUB *(Forts.)*	
Feldname in MQSD	**Besondere Hinweise**
SubCorrelId und PubAccountingToken	Wenn Sie MQSO_SET_CORREL_ID nicht verwenden, generiert der Warteschlangenmanager einen Wert für *SubCorrelId*. Wenn Sie MQSO_SET_IDENTITY_CONTEXT nicht verwenden, generiert der Warteschlangenmanager einen Wert für *PubAccountingToken*. Diese Felder werden im MQSD eines MQSUB-Aufruf mit der Option MQSO_RESUME zurückgegeben. Wenn sie vom Warteschlangenmanager generiert werden, wird der generierte Wert in einem MQSUB-Aufruf mit der Option MQSO_CREATE oder MQSO_ALTER zurückgegeben.
PubPriority, SubLevel & PubApplIdentityData	Diese Felder werden im MQSD zurückgegeben.
ResObjectString	Dieses Nur-Ausgabe-Feld wird im MQSD zurückgegeben, wenn ein Puffer bereitgestellt wird.

C-Aufruf

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   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.
```

Aufruf in PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQSUB,(HCONN,SUBDESC,HOBJ,HSUB,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Subskriptionsanforderung

Verwenden Sie den Aufruf MQSUBRQ, um eine Anforderung für die ständige Veröffentlichung zu stellen, wenn der Subskribent mit MQSO_PUBLICATIONS_ON_REQUEST registriert wurde.

Syntax

MQSUBRQ (*Hconn*, *Hsub*, *Action*, *SubRqOpts*, *Compcode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager. Der Wert von *Hconn* wurde von einem früheren MQCONN- oder MQCONNX-Aufruf zurückgegeben.

Für CICS-Anwendungen unter z/OS und Anwendungen unter IBM i, die im Kompatibilitätsmodus ausgeführt werden, kann der MQCONN-Aufruf übergangen und folgender Wert für *Hconn* angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Hsub
Typ: MQHOBJ - Eingabe

Diese Kennung steht für die Subskription, für die eine Aktualisierung angefordert werden soll. Der Wert von *Hsub* wurde aus einem vorherigen MQSUB-Aufruf zurückgegeben.

Aktion
Typ: MQLONG - Eingabe

Dieser Parameter bestimmt die Aktion, die zur Anwendung auf die Subskription angefordert wird. Der folgende Wert muss angegeben werden:

MQSR_ACTION_PUBLICATION

Diese Aktion fordert an, dass eine Aktualisierungsveröffentlichung für das angegebene Thema gesendet wird. Sie kann nur verwendet werden, wenn der Subskribent die Option MQSO_PUBLICATIONS_ON_REQUEST im MQSUB-Aufruf angegeben hat, als die Subskription erstellt wurde. Verfügt der Warteschlangenmanager über eine ständige Veröffentlichung für das Thema, wird sie an den Abonnenten gesendet. Wenn nicht, schlägt der Aufruf fehl. Wenn eine Anwendung eine ständige Veröffentlichung erhält, wird durch die Nachrichteneigenschaft MQIsRetained der betreffenden Veröffentlichung darauf hingewiesen.

Da das Thema in der vorhandenen Subskription, das durch den Parameter Hsub dargestellt wird, Platzhalter enthalten kann, empfängt der Subskribent möglicherweise mehrere ständige Veröffentlichungen.

SubRqOpts
Typ: MQSRO - Ein-/Ausgabe

Diese Optionen steuern die Aktion von MQSUBRQ, weitere Informationen finden Sie unter „MQSRO - Optionen Subskriptionsanforderung” auf Seite 578.

Wenn keine Optionen erforderlich sind, können Programme, die in C oder S/390-Assembler geschrieben wurden, eine Nullparameteradresse angeben anstatt die Adresse einer MQSRO-Struktur anzugeben.

CompCode
Typ: MQLONG - Ausgabe

Der Beendigungscode; dies ist einer der folgenden Codes:

MQCC_OK
Erfolgreiche Ausführung.

MQCC_WARNING
Warnung (teilweise Ausführung).

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Der Ursachencode, der den *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') Die angeforderte Funktion ist in der aktuellen Umgebung nicht verfügbar.

MQRC_NO_RETAINED_MSG
2437 (X'0985') Derzeit sind keine ständigen Veröffentlichungen für dieses Thema gespeichert.

MQRC_OPTIONS_ERROR
2046 (X'07FE') Optionsparameter oder Feld enthält ungültige Optionen oder eine ungültige Kombination von Optionen.

MQRC_Q_MGR_QUIESCING
2161, (X'0871') Warteschlangenmanager wird in Quiescemodus versetzt.

MQRC_SRO_ERROR
2438 (X'0986') Im MQSUBRQ-Aufruf ist die Subskriptionsanforderungsoption MQSRO nicht gültig.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Ständige Veröffentlichungen, die für die subskribierte Themenzeichenfolge vorhanden sind, können nicht abgerufen werden.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Die ständigen Veröffentlichungen, die für die subskribierte Themenzeichenfolge vorhanden sind, können nicht an die Zielwarteschlange der Subskription und nicht an die Warteschlange für nicht zustellbare Nachrichten übermittelt werden.

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

Hinweise zur Verwendung

Die folgenden Hinweise zur Verwendung gelten für die Verwendung des Aktionscodes MQSR_ACTION_PUBLICATION:

1. Wenn dieses Verb erfolgreich ausgeführt wird, wurden die ständigen Veröffentlichungen, die der angegebenen Subskription entsprechen, an die Subskription gesendet und können unter Verwendung von MQGET oder MQCB mit der Kennung "Hobj" empfangen werden, die im ursprünglichen MQSUB-Verb zurückgegeben wurde, das die Subskription erstellt hat.
2. Wenn das von dem ursprünglichen MQSUB-Verb, das die Subskription erstellt hat, abonnierte Thema einen Platzhalter enthielt, können mehrere ständige Veröffentlichungen gesendet werden. Die Anzahl der als Ergebnis dieses Aufrufs gesendeten Veröffentlichungen wird im Feld NumPubs in der SubRqOpts-Struktur dokumentiert.
3. Wenn dieses Verb mit dem Ursachencode MQRC_NO_RETAINED_MSG ausgeführt wird, lagen für das angegebene Thema derzeit keine ständigen Veröffentlichungen vor.#
4. Wenn dieses Verb mit dem Ursachencode MQRC_RETAINED_MSG_Q_ERROR oder MQRC_RETAINED_NOT_DELIVERED ausgeführt wird, liegen ständige Veröffentlichungen für das Thema vor. Es ist jedoch ein Fehler aufgetreten, der angibt, dass die Veröffentlichungen nicht übergeben werden konnten.
5. Bevor die Anwendung diesen Aufruf ausführen kann, muss sie über eine aktuelle Subskription für das Thema verfügen. Wenn die Subskription in einer früheren Instanz der Anwendung ausgeführt wurde und keine gültige Kennung für die Subskription verfügbar ist, muss die Anwendung zuerst MQSUB mit der Option MQSO_RESUME aufrufen, um eine Kennung für die Verwendung in diesem Aufruf abzurufen.
6. Die Veröffentlichungen werden zu dem Ziel gesendet, das für die Verwendung mit der aktuellen Subskription dieser Anwendung registriert ist. Wenn die Veröffentlichungen an ein anderes Ziel gesendet werden, muss die Subskription zunächst mit dem Aufruf MQSUB mit der Option MQSO_ALTER geändert werden.

C-Aufruf

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf in COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
** 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.
```

Aufruf in PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

Aufruf von High Level Assembler

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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
```

Attribute von Objekten

In dieser Themensammlung sind nur die WebSphere MQ-Objekte aufgelistet, die Gegenstand eines MQINQ-Funktionsaufrufs sein können. Außerdem enthält sie Details zu den Attributen, die abgefragt werden können, und zu den zu verwendenden Selektoren.

Attribute für den Warteschlangenmanager

Einige WS-Manager-Attribute sind für bestimmte Implementierungen festgelegt, andere können mit dem MQSC-Befehl ALTER QMGR geändert werden.

Die Attribute können auch mit dem Befehl DISPLAY QMGRangezeigt werden. Die meisten Warteschlangenmanagerattribute können abgefragt werden, indem ein spezielles MQOT_Q_MGR-Objekt geöffnet und der MQINQ-Aufruf mit der zurückgegebenen Kennung verwendet wird.

Die folgende Tabelle enthält eine Zusammenfassung der Attribute, die für den Warteschlangenmanager spezifisch sind. Die Attribute werden in alphabetischer Reihenfolge beschrieben.

Anmerkung: Die Namen der in diesem Abschnitt erläuterten Attribute sind beschreibende Namen, die mit dem MQINQ-Aufruf verwendet werden; es sind dieselben Namen wie für die PCF-Befehle. Wenn Attribute mit MQSC-Befehlen definiert, geändert oder angezeigt werden, werden alternative Kurznamen verwendet. Weitere Informationen finden Sie unter Scriptbefehle (MQSC).

Tabelle 572. Attribute für den Warteschlangenmanager.
Liste der Warteschlangenmanagerattribute mit Links und Kurzbeschreibung

Attribut	Beschreibung
AccountingConnOverride	Abrechnungseinstellungen überschreiben
AccountingInterval	Gibt an, wie oft temporäre Abrechnungssätze geschrieben werden sollen
ActivityConnOverride	Aktivitätseinstellungen werden überschrieben
ActivityTrace	Steuert die Erfassung des WebSphere MQ MQI-Anwendungsaktivitätstrace
AdoptNewMCACheck	Überprüfte Elemente, um zu ermitteln, ob ein neuer Nachrichtenkanalagent angenommen wird
AdoptNewMCAType	Gibt an, ob eine verwaiste Instanz eines Nachrichtenkanalagenten eines bestimmten Kanaltyps automatisch erneut gestartet wird
AlterationDate	Datum der letzten Änderung der Definition

Tabelle 572. Attribute für den Warteschlangenmanager.
Liste der Warteschlangenmanagerattribute mit Links und Kurzbeschreibung
(Forts.)

Attribut	Beschreibung
AlterationTime	Uhrzeit der letzten Änderung der Definition
AuthorityEvent	Legt fest, ob Berechtigungsereignisse ("Not Authorized") generiert werden
BridgeEvent	Steuerattribut für Bridge-Ereignisse
ChannelAutoDef	Legt fest, ob die automatische Kanaldefinition zulässig ist
ChannelAutoDefEvent	Legt fest, ob die Ereignisse "Channel Automatic-Definition" generiert werden
ChannelAutoDefExit	Name des Benutzerexits für die automatische Kanaldefinition
ChannelEvent	Steuerattribut für Kanalereignisse
ChannelInitiatorControl	Steuerattribut für Kanalinitiator
ChannelMonitoring	Onlineüberwachungsdaten für Kanäle
ChannelStatistics	Steuert die Erfassung statistischer Daten für Kanäle
ChinitAdapters	Anzahl Adapter-Subtasks für die Verarbeitung von WebSphere MQ-Aufrufen
ChinitDispatchers	Anzahl zu verwendender Dispatcher für den Kanalinitiator
	Für Verwendung durch IBM reserviert
ChinitTraceAutoStart	Gibt an, ob der Trace für den Kanalinitiator automatisch gestartet werden soll
ChinitTraceTableSize	Größe des Tracedatenspeicherbereichs des Kanalinitiators
ClusterSenderMonitoringDefault	Standardwert für Onlineüberwachungsdaten für Clustersenderkanäle
ClusterSenderStatistics	Steuert die Erfassung statistischer Überwachungsdaten für Clustersenderkanäle
ClusterWorkloadData	Benutzerdaten für den Exit für Clusterauslastung
ClusterWorkloadExit	Name des Benutzerexits für das Clusterauslastungsmanagement
ClusterWorkloadLength	Maximale Länge der Nachrichtendaten, die an den Exit für Clusterauslastung übergeben werden
CLWLMRUChannels	Anzahl der zuletzt verwendeten Kanäle für Clusterlastausgleich
CLWLUseQ	Clusterauslastung verwendet ferne Warteschlange
CodedCharSetId	ID des codierten Zeichensatzes
CommandEvent	Steuerattribut für Befehlsereignisse
Attribut CommandInputQName	Name der Befehlseingabewarteschlange
CommandLevel	Befehlsebene
Attribut CommandServerControl	Steuerattribut für Befehlsserver
Attribut ConfigurationEvent	Steuerattribut für Konfigurationsereignisse
DeadLetterQName	Name der Warteschlange für nicht zustellbare Nachrichten
DEFCLXQ	Typ der Standard-Clusterübertragungswarteschlange
DefXmitQName	Name der standardmäßigen Übertragungswarteschlange
DistLists	Unterstützung Verteilerliste
DNSGroup	Name der Gruppe für TCP-Empfangsprogramm, wenn Workload Manager Dynamic Domain Name Services verwendet werden
DNSWLM	Gibt an, ob TCP-Empfangsprogramm bei Workload Manager Dynamic Domain Name Services registriert wird
ExpiryInterval	Intervall zwischen Überprüfungen auf abgelaufene Nachrichten
IGQPutAuthority	PUT-Berechtigung für gruppeninterne Warteschlangensteuerung
IGQUserId	Benutzer-ID der gruppeninternen Warteschlangensteuerung
InhibitEvent	Legt fest, ob Inhibit-Ereignisse ("Inhibit Get" und "Inhibit Put") generiert werden
IPAddressVersion	Version der Internet Protocol-Adresse
IntraGroupQueuing	Unterstützung für gruppeninterne Warteschlangensteuerung
ListenerTimer	Zeitintervall zwischen den Versuchen, das Empfangsprogramm nach APPC- oder TCP/IP-Fehler erneut zu starten

Tabelle 572. Attribute für den Warteschlangenmanager.

Liste der Warteschlangenmanagerattribute mit Links und Kurzbeschreibung

(Forts.)

Attribut	**Beschreibung**
LocalEvent	Legt fest, ob lokale Fehlerereignisse generiert werden
LoggerEvent	Steuert, ob Protokollierungsereignisse generiert werden
LUGroupName	Generischer LU-Name für das LU 6.2-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet
LUName	Name der für ausgehende LU 6.2-Übertragungen zu verwendenden LU
LU62ARMSuffix	Suffix des SYS1.PARMLIB-Mitglieds APPCPMxx, das die LUADD für diesen Kanalinitiator benennt
LU62Channels	Maximale Anzahl aktueller Kanäle oder verbundener Clients, die LU 6.2. verwenden
MaxActiveChannels	Maximale Anzahl Kanäle, die zu jeder Zeit aktiv sein können
MaxChannels	Maximale Anzahl derzeit aktiver Kanäle
MaxHandles	Maximale Anzahl von Kennungen
MaxMsgLength	Maximale Nachrichtenlänge in Byte
Attribut MaxPriority	Maximale Priorität
MaxPropertiesLength	Maximale Länge von Eigenschaftsdaten in Byte
MaxUncommittedMsgs	Maximale Anzahl nicht festgeschriebener Nachrichten in einer Arbeitseinheit
MQIAccounting	Steuert die Erfassung von Abrechnungsinformationen für MQI-Daten
MQIStatistics	Steuert die Erfassung statistischer Überwachungsdaten für Warteschlangenmanager
MsgMarkBrowseInterval	Intervall, nachdem der Warteschlangenmanager die Markierung von durchsuchten Nachrichten entfernen kann
OutboundPortMin	Legt zusammen mit *OutboundPortMin* den Bereich der Portnummern fest, die bei der Bindung abgehender Kanäle verwendet werden sollen
OutboundPortMin	Legt zusammen mit *OutboundPortMax* den Bereich der Portnummern fest, die bei der Bindung abgehender Kanäle verwendet werden sollen
PerformanceEvent	Legt fest, ob leistungsspezifische Ereignisse generiert werden
Plattform	Plattform, auf der der Warteschlangenmanager ausgeführt wird
PubSubNPInputMsg	Gibt an, ob eine nicht zugestellte Nachricht gelöscht (oder beibehalten) wird
PubSubNPResponse	Steuert das Verhalten nicht zugestellter Nachrichten
PubSubMaxMsgRetryCount	Anzahl Wiederholungen bei der Verarbeitung (unter Synchronisationspunkt) einer fehlgeschlagenen Befehlsnachricht
PubSubSyncPoint	Gibt an, ob nur persistente (oder alle) Nachrichten unter Synchronisationspunkt verarbeitet werden sollen
PubSubMode	Gibt an, ob die Publish/Subscribe-Schnittstelle ausgeführt wird
QMgrDesc	Beschreibung des Warteschlangenmanagers
QMgrIdentifier	Eindeutige intern generierte ID des Warteschlangenmanagers
QMgrName	Name des Warteschlangenmanagers
QSGName	Name der Gruppe mit gemeinsamer Warteschlange
QueueAccounting	Steuert die Erfassung von Abrechnungsinformationen für Warteschlangen
QueueMonitoring	Onlineüberwachungsdaten für Warteschlangen
QueueStatistics	Steuert die Erfassung von Statistikdaten für Warteschlangen
ReceiveTimeout	Gibt an, wie lange der TCP/IP-Kanal auf Daten wartet, bevor er in den inaktiven Zustand zurückkehrt
ReceiveTimeoutMin	Qualifikationsmerkmal für *ReceiveTimeout*
ReceiveTimeoutType	Mindestzeit, die der TCP/IP-Kanal auf Daten wartet, bevor er in den inaktiven Zustand zurückkehrt
RemoteEvent	Legt fest, ob ferne Fehlerereignisse generiert werden
RepositoryName	Gibt den Namen des Clusters an, für den dieser Warteschlangenmanager Repository-Services bereitstellt
RepositoryNamelist	Gibt den Namen des Namenslistenobjekts an, das die Namen von Clustern enthält, für die dieser Warteschlangenmanager Repository-Services bereitstellt

Tabelle 572. Attribute für den Warteschlangenmanager.
Liste der Warteschlangenmanagerattribute mit Links und Kurzbeschreibung
(Forts.)

Attribut	Beschreibung
ScyCase	Groß-/Kleinschreibung von Sicherheitsprofilen
SharedQMgrName	Warteschlangenmanagername der gemeinsam genutzten Warteschlange
„SPLCAP" auf Seite 829	WebSphere MQ Advanced Message Security-Schutz für einen Warteschlangenmanager aktiviert oder inaktiviert
SSLCRLNamelist 1	Name des Namenslistenobjekts mit Namen von Authentifizierungsdatenobjekten
SSLCryptoHardware 1	Konfigurationszeichenfolge für Verschlüsselungshardware
SSLEvent	Steuerattribut für SSL-Ereignisse
SSLFIPSRequired	Nur FIPS-zertifizierte Algorithmen für Verschlüsselung verwenden
SSLKeyRepository 1	Speicherposition des SSL-Schlüsselrepository
SSLKeyResetCount	Rücksetzungszähler für SSL-Schlüssel
SSLTasks 1	Anzahl Serversubtasks für die Verarbeitung von SSL-Aufrufen
StatisticsInterval	Gibt an, wie oft statistische Überwachungsdaten geschrieben werden sollen
StartStopEvent	Legt fest, ob Start- und Stoppereignisse generiert werden
SyncPoint	Synchronisationspunktverfügbarkeit
TCPChannels	Maximale Anzahl aktueller Kanäle oder verbundener Clients, die TCP/IP verwenden
TCPKeepAlive	Gibt an, ob TCP KEEPALIVE verwendet wird, um das andere Ende der Verbindung zu überprüfen
TCPName	Name des verwendeten TCP/IP-Systems
TCPStackType	Gibt an, wie der Kanalinitiator TCP/IP-Adressen verwenden kann
Attribut TraceRouteRecording	Steuert die Aufzeichnung von Traceroute-Informationen
TriggerInterval	Intervall der Auslösenachricht
Version	Version
XrCapability	Gibt an, ob Telemetry-Befehle unterstützt werden.

Anmerkungen:

1. Dieses Attribut kann nicht mit dem MQINQ-Aufruf abgefragt werden und ist in diesem Abschnitt nicht beschrieben. Weitere Einzelheiten zu diesem Attribut finden Sie unter Warteschlangenmanager ändern.

Zugehörige Tasks

Angeben, dass nur FIPS-zertifizierte CipherSpecs während der Ausführung auf dem MQI-Client verwendet werden

Zugehörige Verweise

Federal Information Processing Standards (FIPS) für UNIX, Linux und Windows

AccountingConnOverride (MQLONG)

Dieses Attribut ermöglicht es Anwendungen, die Einstellung der Werte von ACCTMQI und ACCTQDATA im Warteschlangenmanagerattribut zu überschreiben.

Folgende Werte sind möglich:

MQMON_DISABLED
Anwendungen können die Einstellung der Warteschlangenmanagerattribute ACCTMQI und ACCTQ mit den Optionsfeldern in der MQCNO-Struktur im Aufruf MQCONNX nicht überschreiben. Dies ist der Standardwert.

MQMON_ENABLED
Anwendungen können die Warteschlangenmanagerattribute ACCTQ und ACCTMQI mit den Optionsfeldern in der MQCNO-Struktur überschreiben.

Änderungen dieser Werte sind nur für Verbindungen zum Warteschlangenmanager wirksam, die nach den Änderungen hergestellt werden.

Dieses Attribut wird nur auf IBM i-, Unix- und Windows-Systemen unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACCOUNTING_CONN_OVERRIDE im MQINQ-Aufruf bestimmt.

AccountingInterval (MQLONG)

Dieses Attribut gibt an, wie oft temporäre Abrechnungsdatensätze geschrieben werden (in Sekunden).

Der Wert ist eine Ganzzahl im Bereich von 0 bis 604800, mit einem Standardwert von 1800 (30 Minuten). Geben Sie 0 an, um temporäre Datensätze zu inaktivieren.

Dieses Attribut wird nur auf IBM i-, Windows-, UNIX- und Linux-Systemen unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACCOUNTING_INTERVAL im MQINQ-Aufruf bestimmt.

ActivityConnOverride (MQLONG)

Mit diesem Attribut können Anwendungen die Einstellung des Werts ACTVTRC im Warteschlangenmanagerattribut überschreiben.

Folgende Werte sind möglich:

MQMON_DISABLED
Anwendungen können die Einstellung des Warteschlangenmanagerattributs ACTVTRC nicht mit den Optionsfeldern der MQCNO-Struktur im Aufruf MQCONNX überschreiben. Dies ist der Standardwert.

MQMON_ENABLED
Anwendungen können das Warteschlangenmanagerattribut ACTVTRC mit den Optionsfeldern der MQCNO-Struktur im Aufruf MQCONNX überschreiben.

Änderungen dieser Werte sind nur für Verbindungen zum Warteschlangenmanager wirksam, die nach den Änderungen hergestellt werden.

Dieses Attribut wird nur auf IBM i-, Unix- und Windows-Systemen unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACTIVITY_CONN_OVERRIDE im Aufruf MQINQ bestimmt.

ActivityTrace (MQLONG)

Dieses Attribut steuert die Erfassung des WebSphere MQ MQI-Anwendungsaktivitätstrace.

Folgende Werte sind möglich:

MQMON_ON
Das WebSphere MQ MQI-Anwendungsaktivitätstrace wird erfasst.

MQMON_OFF
Das WebSphere MQ MQI-Anwendungsaktivitätstrace wird nicht erfasst. Dies ist der Standardwert.

Wenn Sie das Warteschlangenmanagerattribut ACTVCONO auf ENABLED setzen, wird dieser Wert unter Umständen für einzelne Verbindungen mit dem Optionsfeld in der MQCNO-Struktur überschrieben.

Änderungen dieser Werte sind nur für Verbindungen zum Warteschlangenmanager wirksam, die nach den Änderungen hergestellt werden.

Dieses Attribut wird nur auf IBM i-, Unix- und Windows-Systemen unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACTIVITY_TRACE im Aufruf MQINQ bestimmt.

AdoptNewMCACheck (MQLONG)

Dieses Attribut legt fest, welche Elemente überprüft werden, um zu ermitteln, ob ein Nachrichtenkanalagent angenommen wird, wenn ein neuer eingehender Kanal erkannt wird, der denselben Namen wie ein bereits aktiver Nachrichtenkanalagent hat

Folgende Werte sind möglich:

MQADOPT_CHECK_Q_MGR_NAME
Der Name des Warteschlangenmanagers wird überprüft.

MQADOPT_CHECK_NET_ADDR
Die Netzadresse wird überprüft.

MQADOPT_CHECK_ALL
Der Warteschlangenmanagername und die Netzadresse werden überprüft. Wenn möglich, sollten Sie diese Überprüfung durchführen, um Ihre Kanäle vor versehentlichem oder böswilligen Beenden zu schützen. Dies ist der Standardwert.

MQADOPT_CHECK_NONE
Keine Elemente überprüfen.

Änderungen dieses Attributs werden wirksam, wenn ein Kanal das nächste Mal versucht, einen Kanal zu übernehmen.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ADOPTNEWMCA_CHECK im MQINQ-Aufruf bestimmt.

AdoptNewMCAType (MQLONG)

Dieses Attribut gibt an, ob eine verwaiste Instanz eines Nachrichtenkanalagenten eines bestimmten Kanaltyps automatisch erneut gestartet wird, wenn eine neue eingehende Kanalanforderung erkannt wird, die dem Attribut AdoptNewMCACheck entspricht.

Folgende Werte sind möglich:

MQADOPT_TYPE_NO
Die Übernahme verwaister Kanalinstanzen ist nicht erforderlich. Dies ist der Standardwert.

MQADOPT_TYPE_ALL
Es werden alle Kanaltypen übernommen.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ADOPTNEWMCA_TYPE im MQINQ-Aufruf bestimmt.

AlterationDate (MQCHAR12)

Dieses Attribut gibt das Datum an, an dem die Definition zuletzt geändert wurde. Das Format des Datums ist YYYY-MM-DDund wird mit zwei abschließenden Leerzeichen aufgefüllt, damit die Länge 12 Byte beträgt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_DATE im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_DATE_LENGTH vorgegeben.

AlterationTime (MQCHAR8)

Dieses Attribut gibt die Uhrzeit an, zu der die Definition zuletzt geändert wurde. Das Zeitformat lautet HH.MM.SS.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_TIME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_TIME_LENGTH vorgegeben.

AuthorityEvent (MQLONG)

Dieses Attribut steuert, ob Autorisierungsereignisse (Not Authorized) generiert werden. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_AUTHORITY_EVENT im MQINQ-Aufruf bestimmt.

BridgeEvent (MQLONG)

Dieses Attribut gibt an, ob IMS-Brückenereignisse generiert werden.

Folgende Werte sind möglich:

MQEVR_ENABLED
IMS-Brückenereignisse folgendermaßen generieren:

MQRC_BRIDGE_STARTED
MQRC_BRIDGE_STOPPED

MQEVR_DISABLED
Keine IMS-Brückenereignisse generieren. Dies ist der Standardwert.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_BRIDGE_EVENT im MQINQ-Aufruf bestimmt.

ChannelAutoDef (MQLONG)

Dieses Attribut steuert die automatische Definition von Kanälen des Typs MQCHT_RECEIVER und MQCHT_SVRCONN. Die automatische Definition von MQCHT_CLUSSDR-Kanälen ist immer aktiviert. Folgende Werte sind möglich:

MQCHAD_DISABLED
Automatische Definition von Kanälen inaktiviert.

MQCHAD_ENABLED
Automatische Definition von Kanälen aktiviert.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris und Windowsunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHANNEL_AUTO_DEF im MQINQ-Aufruf bestimmt.

ChannelAutoDefEvent (MQLONG)

Dieses Attribut gibt an, ob 'Automatische Kanaldefinition'-Ereignisse generiert werden. Gilt nur für Kanäle des Typs MQCHT_RECEIVER, MQCHT_SVRCONN und MQCHT_CLUSSDR. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris und Windowsunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHANNEL_AUTO_DEF_EVENT im MQINQ-Aufruf bestimmt.

ChannelAutoDefExit (MQCHARn)

Dieses Attribut gibt den Namen des Benutzerexits für automatische Kanaldefinition an. Wenn dieser Name belegt ist und *ChannelAutoDef* den Wert MQCHAD_ENABLED hat, wird der Exit jedes Mal angerufen, wenn der Warteschlangenmanager eine Kanaldefinition erstellt. Dies gilt für Kanäle des Typs MQCHT_RECEIVER, MQCHT_SVRCONN und MQCHT_CLUSSDR. Der Exit kann dann eine der folgenden Aktionen durchführen:

- Kanaldefinition ohne Änderungen erstellen
- Attribute der Kanaldefinition ändern, die erstellt wird
- Erstellung des Kanals vollständig unterdrücken

Anmerkung: Sowohl die Länge als auch der Wert dieses Attributs sind umgebungsspezifisch. In der Einführung in die MQCD-Struktur unter „MQCD - Kanaldefinition” auf Seite 1058 finden Sie Informationen zum Wert dieses Attributs in verschiedenen Umgebungen.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windowsund z/OSunterstützt. Unter z/OS gilt es nur für Clustersender- und Clusterempfängerkanäle.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CHANNEL_AUTO_DEF_EXIT im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_EXIT_NAME_LENGTH angegeben.

ChannelEvent (MQLONG)
Gibt an, ob Kanalereignisse generiert werden.

Folgende Werte sind möglich:

MQEVR_EXCEPTION
Nur die folgenden Kanalereignisse generieren:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED, wobei ReasonQualifier gesetzt ist auf:

  MQRQ_CHANNEL_STOPPED_ERROR
  MQRQ_CHANNEL_STOPPED_RETRY
  MQRQ_CHANNEL_STOPPED_DISABLED

  MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED
Alle Kanalereignisse generieren, d. h., zusätzlich zu den durch EXCEPTION generierten folgende Kanalereignisse generieren:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED, wobei ReasonQualifier gesetzt ist auf:

  MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED
Keine Kanalereignisse generieren. Dies ist der Standardwert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHANNEL_EVENT im MQINQ-Aufruf bestimmt.

ChannelInitiatorControl (MQLONG)
Gibt an, ob der Kanalinitiator beim Start des Warteschlangenmanagers gestartet werden soll.

Folgende Werte sind möglich:

MQSVC_CONTROL_MANUAL
Der Kanalinitiator wird nicht automatisch gestartet.

MQSVC_CONTROL_Q_MGR
Der Kanalinitiator soll beim Start des Warteschlangenmanagers automatisch gestartet werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHINIT_CONTROL im MQINQ-Aufruf bestimmt.

ChannelMonitoring (MQLONG)
Dieses Attribut gibt die Onlineüberwachungsdaten für Kanäle an.

Folgende Werte sind möglich:

MQMON_NONE
Datenerfassung für Kanalüberwachung bei allen Kanälen inaktivieren, unabhängig von der Einstellung des Kanalattributs MONCHL. Dies ist der Standardwert.

MQMON_OFF
Überwachungsdatenerfassung bei Kanälen ausschalten, bei denen QMGR im Kanalattribut MONCHL angegeben ist.

MQMON_LOW
Überwachungsdatenerfassung mit niedriger Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut MONCHL angegeben ist.

MQMON_MEDIUM
Überwachungsdatenerfassung mit mittlerer Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut MONCHL angegeben ist.

MQMON_HIGH
Überwachungsdatenerfassung mit hoher Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut MONCHL angegeben ist.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MONITORING_CHANNEL im MQINQ-Aufruf bestimmt.

ChannelStatistics (MQLONG)

Dieses Attribut steuert die Erfassung von Statistikdaten für Kanäle.

Folgende Werte sind möglich:

MQMON_NONE
Datenerfassung für Kanalstatistik bei allen Kanälen inaktivieren, unabhängig von der Einstellung des Kanalattributs STATCHL. Dies ist der Standardwert.

MQMON_OFF
Erfassung statistischer Daten bei Kanälen ausschalten, bei denen QMGR im Kanalattribut STATCHL angegeben ist.

MQMON_LOW
Erfassung statistischer Daten mit niedriger Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut STATCHL angegeben ist.

MQMON_MEDIUM
Erfassung statistischer Daten mit mittlerer Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut STATCHL angegeben ist.

MQMON_HIGH
Erfassung statistischer Daten mit hoher Erfassungsrate bei Kanälen aktivieren, bei denen QMGR im Kanalattribut STATCHL angegeben ist.

Für die meisten Systeme wird empfohlen, die Einstellung MEDIUM zu verwenden. Bei einem Kanal, der hohe Nachrichtenvolumen pro Sekunde verarbeitet, empfiehlt sich möglicherweise die Erfassungsstufe LOW. Die Auswahl von HIGH empfiehlt sich bei einem Kanal, der nur wenige Nachrichten verarbeitet und bei dem die aktuellsten Informationen wichtig sind.

Dieses Attribut wird nur unter IBM i, UNIX-Systemen und Windows unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_STATISTICS_CHANNEL im MQINQ-Aufruf bestimmt.

ChinitAdapters (MQLONG)

Dieses Attribut gibt die Anzahl Adaptersubtasks für die Verarbeitung von WebSphere MQ-Aufrufen an. Der Wert muss 0-9999 sein, der Standardwert ist 8.

Das Verhältnis von Adaptern zu Dispatchern (Attribut "ChinitDispatchers") sollte etwa 8 bis 5 betragen. Wenn Sie jedoch nur wenige Kanäle haben, müssen Sie den Wert dieses Parameters nicht aus dem

Standardwert herabsetzen. Sie können die folgenden Werte verwenden: Testsystem 8 (Standard), Produktionssystem 20. Idealerweise sollten Sie über 20 Adapter verfügen, um die Parallelverarbeitung von WebSphere MQ-Aufrufen optimal nutzen zu können. Dies ist vor allem bei persistenten Nachrichten wichtig. Bei nicht persistenten Nachrichten können weniger Adapter von Vorteil sein.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHINIT_ADAPTERS im MQINQ-Aufruf bestimmt.

ChinitDispatchers (MQLONG)

Dieses Attribut gibt die Anzahl Dispatcher an, die für den Kanalinitiator verwendet werden sollen. Der Wert muss 0-9999 sein, der Standardwert ist 5.

Als Richtlinie gilt, dass ein einzelner Dispatcher für 50 aktive Kanäle verwendet wird. Wenn Sie jedoch nur über wenige Kanäle verfügen, können Sie den Standardwert verwenden und müssen den Wert dieses Attributs nicht verringern. Bei Verwendung von TCP/IP liegt die maximale Anzahl Dispatcher für TCP/IP-Kanäle bei 100, auch wenn Sie einen höheren Wert angeben. Sie können die folgenden Einstellungen verwenden: Testsysteme 5 (Standardwert), Produktionssysteme 20 (Sie benötigen 20 Dispatcher, um 1000 aktive Kanäle zu verarbeiten).

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHINIT_DISPATCHERS im MQINQ-Aufruf bestimmt.

ChinitTraceAutoStart (MQLONG)

Dieses Attribut gibt an, ob der Kanalinitiatortrace automatisch gestartet werden soll.

Folgende Werte sind möglich:

MQTRAXSTR_YES
Kanalinitiatortrace automatisch starten. Dies ist der Standardwert.

MQTRAXSTR_NO
Kanalinitiatortrace nicht automatisch starten.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHINIT_TRACE_AUTO_START im MQINQ-Aufruf bestimmt.

ChinitTraceTableSize (MQLONG)

Dieses Attribut gibt die Größe des Tracedatenspeicherbereichs des Kanalinitiators an (in MB).

Der Wert muss im Bereich von 0 bis 2048 liegen, der Standardwert ist 2.

Anmerkung: Wenn Sie umfangreiche z/OS-Datenspeicherbereiche verwenden, stellen Sie sicher, dass ausreichend Zusatzspeicher auf Ihrem System vorhanden ist, um alle zugehörigen z/OS-Auslagerungsaktivitäten zu unterstützen. Möglicherweise müssen Sie auch die Speicherauszugsdatei SYS1.DUMP vergrößern.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CHINIT_TRACE_TABLE_SIZE im MQINQ-Aufruf bestimmt.

ClusterSenderMonitoringDefault (MQLONG)

Gibt den Wert an, der für das Attribut "ChannelMonitoring" der automatisch definierten Clustersenderkanäle ersetzt werden soll.

Folgende Werte sind möglich:

MQMON_Q_MGR
Die Einstellung für die Erfassung von Onlineüberwachungsdaten wird vom Warteschlangenmanagerattribut *ChannelMonitoring* übernommen. Dies ist der Standardwert.

MQMON_OFF
Die Kanalüberwachung wird inaktiviert.

MQMON_LOW
Wenn für *ChannelMonitoring* ein anderer Wert als MQMON_NONE angegeben wird, wird die Überwachung mit einer geringen Datenerfassungsrate aktiviert, die nur minimale Auswirkungen auf die Systemleistung hat. Die erfassten Daten sind nicht unbedingt die aktuellsten Daten.

MQMON_MEDIUM
Wenn für *ChannelMonitoring* ein anderer Wert als MQMON_NONE angegeben wird, wird die Überwachung mit einer mittleren Datenerfassungsrate aktiviert, die begrenzte Auswirkungen auf die Systemleistung hat.

MQMON_HIGH
Wenn für *ChannelMonitoring* ein anderer Wert als MQMON_NONE angegeben wird, wird die Überwachung mit einer hohen Datenerfassungsrate aktiviert, die wahrscheinlich Auswirkungen auf die Systemleistung hat. Bei den erfassten Daten handelt es sich um die aktuellsten Daten.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MONITORING_AUTO_CLUSSDR im MQINQ-Aufruf bestimmt.

ClusterSenderStatistics (MQLONG)

Da Clustersenderkanäle automatisch aus der Definition von CLUSRCVR im Repository definiert werden können, können Sie die Einstellung des Attributs STATCHL für diese automatisch definierten Clustersenderkanäle nicht mit ALTER CHANNEL ändern. Bei diesen Kanälen basiert die Entscheidung, ob Onlineüberwachungsdaten erfasst werden sollen, auf der Einstellung dieses Warteschlangenmanagerattributs.

Folgende Werte sind möglich:

MQMON_Q_MGR
Die Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle basiert auf dem Wert des Warteschlangenmanagerattributs STATCHL. Dies ist der Standardwert.

MQMON_OFF
Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle deaktivieren.

MQMON_LOW
Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle mit niedriger Erfassungsrate aktivieren.

MQMON_MEDIUM
Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle mit mittlerer Erfassungsrate aktivieren.

MQMON_HIGH
Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle mit hoher Erfassungsrate aktivieren.

Für die meisten Systeme wird empfohlen, die Einstellung MEDIUM zu verwenden. Bei einem automatisch definierten Clustersenderkanal, der hohe Nachrichtenvolumen pro Sekunde verarbeitet, empfiehlt sich möglicherweise die Erfassungsstufe LOW. Die Auswahl von HIGH empfiehlt sich bei einem Kanal, der nur wenige Nachrichten verarbeitet und bei dem die aktuellsten Informationen wichtig sind.

Der Wert dieses Attributs wird mit dem Selektor MQIA_STATISTICS_AUTO_CLUSSDR im MQINQ-Aufruf bestimmt.

ClusterWorkloadData (MQCHAR32)

Es handelt sich um eine benutzerdefinierte 32-Byte-Zeichenfolge, die beim Aufruf des Exits für Clusterauslastung an diesen übergeben wird. Wenn keine Daten zum Übergeben an den Exit vorhanden sind, ist die Zeichenfolge leer.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windows und z/OSunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CLUSTER_WORKLOAD_DATA im MQINQ-Aufruf bestimmt.

ClusterWorkloadExit (MQCHARn)

Dies ist der Name des Benutzerexits für das Management der Clusterauslastung. Wenn dieser Name nicht leer ist, wird der Exit jedes Mal aufgerufen, wenn eine Nachricht in eine Clusterwarteschlange eingereiht oder von einer Clustersenderwarteschlange zu einer anderen verschoben wird. Der Exit kann die vom Warteschlangenmanager als Ziel für die Nachricht ausgewählte Warteschlangeninstanz akzeptieren oder eine andere Warteschlangeninstanz auswählen.

Anmerkung: Sowohl die Länge als auch der Wert dieses Attributs sind umgebungsspezifisch.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windows und z/OSunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CLUSTER_WORKLOAD_EXIT im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_EXIT_NAME_LENGTH angegeben.

ClusterWorkloadLength (MQLONG)

Dies ist die maximale Länge der Nachrichtendaten, die an den Exit für Clusterauslastung übergeben werden. Die effektive Länge von Daten, die an den Exit übergeben werden, ergibt das Minimum für folgende Werte:

- Die Länge der Nachricht.
- Das Attribut *MaxMsgLength* des Warteschlangenmanagers.
- Attribut *ClusterWorkloadLength*

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windows und z/OSunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CLUSTER_WORKLOAD_LENGTH im MQINQ-Aufruf bestimmt.

CLWLMRUChannels (MQLONG)

Dies gibt die maximale Anzahl an MRU-Clusterkanälen an, die für die Verwendung durch den Algorithmus zur Auswahl der Clusterauslastung berücksichtigt werden müssen.

Dieser Wert liegt zwischen 1 und 999999999.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CLWL_MRU_CHANNELS im MQINQ-Aufruf bestimmt.

CLWLUseQ (MQLONG)

Dieses Attribut gibt an, ob ferne Warteschlangen für die Clusterauslastung verwendet werden sollen.

Folgende Werte sind möglich:

MQCLWL_USEQ_ANY
Es werden lokale und ferne Warteschlangen verwendet.

MQCLWL_USEQ_LOCAL
Es werden keine fernen Warteschlangen verwendet. Dies ist der Standardwert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CLWL_USEQ im MQINQ-Aufruf bestimmt.

CodedCharSetId (MQLONG)

Das Attribut definiert den Zeichensatz, der vom Warteschlangenmanager für alle im MQI definierten Zeichenfolgefelder verwendet wird, etwa für Erstellungsdatum und -uhrzeit der Warteschlange oder die Namen von Objekten. Der Zeichensatz muss Einzelbytezeichen für die Zeichen verwenden, die in Objektnamen gültig sind. Er gilt nicht für Anwendungsdaten, die in der Nachricht übertragen werden. Der Wert hängt von der Umgebung ab:

- Unter z/OS wird der Wert anhand der Systemparameter festgelegt, wenn der Warteschlangenmanager gestartet wird. Der Standardwert ist 500.
- Unter Windowsist der Wert die primäre CODEPAGE des Benutzers, der den WS-Manager erstellt.
- Unter IBM i entspricht der Wert dem Wert, der bei der ersten Erstellung des Warteschlangenmanagers in der Umgebung festgelegt wird.
- Auf UNIX -Systemen ist der Wert der standardmäßige CODESET für die Ländereinstellung des Benutzers, der den Warteschlangenmanager erstellt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CODED_CHAR_SET_ID im MQINQ-Aufruf bestimmt.

CommandEvent (MQLONG)

Dieses Attribut gibt wie folgt an, ob Befehlsereignisse generiert werden sollen:

MQEVR_DISABLED
Es werden keine Befehlsereignisse generiert. Dies ist die Standardeinstellung.

MQEVR_ENABLED
Es werden Befehlsereignisse generiert.

MQEVR_NO_DISPLAY
Befehlsereignisse werden für alle erfolgreich ausgeführten Befehle (außer dem Befehl MQINQ) generiert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_COMMAND_EVENT im MQINQ-Aufruf bestimmt.

CommandInputQName (MQCHAR48)

Dies ist der Name der Befehlseingabewarteschlange, die im lokalen Warteschlangenmanager definiert ist. Dabei handelt es sich um eine Warteschlange, an die Benutzer Befehle senden können, wenn sie dazu berechtigt sind. Der Name der Warteschlange hängt von der Umgebung ab:

- Unter z/OSlautet der Name der Warteschlange SYSTEM.COMMAND.INPUT; MQSC-und PCF-Befehle können an sie gesendet werden. Im Abschnitt MQSC-Befehle finden Sie Einzelheiten zu MQSC-Befehlen und im Abschnitt Definitionen der PCFs (Programmable Command Formats) finden Sie Informationen zu PCF-Befehlen.
- In allen anderen Umgebungen lautet der Name der Warteschlange SYSTEM.ADMIN.COMMAND.QUEUE, und an diese Warteschlange können nur PCF-Befehle gesendet werden. Ein MQSC-Befehl kann jedoch an diese Warteschlange gesendet werden, wenn der MQSC-Befehl in einem PCF-Befehl vom Typ MQCMD_ESCAPE enthalten ist. Weitere Informationen zum Escape-Befehl finden Sie unter Escape.

Der Wert dieses Attributs wird mit dem Selektor MQCA_COMMAND_INPUT_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

CommandLevel (MQLONG)

Gibt die Ebene der Systemsteuerbefehle an, die vom Warteschlangenmanager unterstützt wird. Folgende Werte sind möglich:

MQCMDL_LEVEL_1
Systemsteuerbefehle Ebene 1.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- MQSeries für AIX Version 2 Release 2
- MQSeries für
  - Version 1 Release 1.1
  - Version 1 Release 1.2
  - Version 1 Release 1.3

- MQSeries für OS/400
  - Version 2 Release 3
  - Version 3 Release 1
  - Version 3 Release 6
- MQSeries für Windows Version 2 Release 0

MQCMDL_LEVEL_101
MQSeries für Windows Version 2 Release 0.1.

MQCMDL_LEVEL_110
MQSeries für Windows Version 2 Release 1.

MQCMDL_LEVEL_114
MQSeries für Version 1 Release 1.4.

MQCMDL_LEVEL_120
MQSeries für Version 1 Release 2.0.

MQCMDL_LEVEL_200
MQSeries für Windows NT Version 2 Release 0.

MQCMDL_LEVEL_210
MQSeries für OS/390 Version 2 Release 1.0.

MQCMDL_LEVEL_220
Systemsteuerbefehle Ebene 220.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- MQSeries für AT & T GIS UNIX Version 2 Release 2
- MQSeries für SINIX und DC/OSx Version 2 Release 2
- MQSeries für SunOS Version 2 Release 2
- MQSeries für Tandem NonStop Kernel Version 2 Release 2

MQCMDL_LEVEL_221
Systemsteuerbefehle Ebene 221.

Dieser Wert wird von MQSeries for AIX Version 2 Release 2.1 zurückgegeben.

MQCMDL_LEVEL_320
Systemsteuerbefehle Ebene 320.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- MQSeries für OS/400
  - Version 3 Release 2
  - Version 3 Release 7

MQCMDL_LEVEL_420
Systemsteuerbefehle Ebene 420.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- MQSeries für IBM i
  - Version 4 Release 2.0
  - Version 4 Release 2.1

MQCMDL_LEVEL_500
Systemsteuerbefehle Ebene 500.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 5 Release 0
- MQSeries für HP-UX Version 5 Release 0

- MQSeries für Solaris Version 5 Release 0
- MQSeries für Windows NT Version 5 Release 0

MQCMDL_LEVEL_510
Systemsteuerbefehle Ebene 510.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 5 Release 1
- MQSeries für AS/400 Version 5 Release 1
- MQSeries für HP-UX Version 5 Release 1
- IBM WebSphere MQ for HP Integrity NonStop Server Version 5 Release 3
- MQSeries for Compaq Tru64 UNIX Version 5 Release 1
- MQSeries für Solaris Version 5 Release 1
- MQSeries für Windows NT Version 5 Release 1

MQCMDL_LEVEL_520
Systemsteuerbefehle Ebene 520.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- MQSeries for AIX Version 5 Release 2
- MQSeries für AS/400 Version 5 Release 2
- MQSeries für HP-UX Version 5 Release 2
- MQSeries für Linux Version 5 Release 2
- MQSeries für OS/390 Version 5 Release 2
- MQSeries für Sun Solaris Version 5 Release 2
- MQSeries für Windows NT Version 5 Release 2

MQCMDL_LEVEL_530
Systemsteuerbefehle Ebene 530.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 5 Release 3
- IBM WebSphere MQ for HP-UX Version 5 Release 3
- IBM WebSphere MQ for i/Series Version 5 Release 3
- IBM WebSphere MQ for Linux für Intel Version 5 Release 3
- IBM WebSphere MQ for Linux für zSeries Version 5 Release 3
- IBM WebSphere MQ for Solaris Version 5 Release 3
- IBM WebSphere MQ for Windows Version 5 Release 3
- IBM WebSphere MQ for z/OS Version 5 Release 3

MQCMDL_LEVEL_600
Systemsteuerbefehle Ebene 600.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 6.0
- IBM WebSphere MQ for HP-UX Version 6.0
- IBM WebSphere MQ für i/Series Version 6.0
- IBM WebSphere MQ for Linux Version 6.0
- IBM WebSphere MQ for Solaris Version 6.0
- IBM WebSphere MQ for Windows Version 6.0
- IBM WebSphere MQ for z/OS Version 6.0

MQCMDL_LEVEL_700
Systemsteuerbefehle Ebene 700.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 7.0
- IBM WebSphere MQ for HP-UX Version 7.0
- IBM WebSphere MQ for IBM i Version 7.0
- IBM WebSphere MQ for Linux Version 7.0
- IBM WebSphere MQ for Solaris Version 7.0
- IBM WebSphere MQ for Windows Version 7.0
- IBM WebSphere MQ for z/OS Version 7.0

MQCMDL_LEVEL_701
Systemsteuerbefehle Ebene 701.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 7.0.1
- IBM WebSphere MQ for HP-UX Version 7.0.1
- IBM WebSphere MQ for IBM i Version 7.0.1
- IBM WebSphere MQ for Linux Version 7.0.1
- IBM WebSphere MQ for Solaris Version 7.0.1
- IBM WebSphere MQ for Windows Version 7.0.1
- IBM WebSphere MQ for z/OS Version 7.0.1

MQCMDL_LEVEL_710
Systemsteuerbefehle Ebene 710.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 7.1
- IBM WebSphere MQ for HP-UX Version 7.1
- IBM WebSphere MQ for IBM i Version 7.1
- IBM WebSphere MQ for Linux Version 7.1
- IBM WebSphere MQ for Solaris Version 7.1
- IBM WebSphere MQ for Windows Version 7.1
- IBM WebSphere MQ for z/OS Version 7.1

MQCMDL_LEVEL_750
Systemsteuerbefehle Ebene 750.

Dieser Wert wird von den folgenden Versionen von IBM WebSphere MQ zurückgegeben:

- IBM WebSphere MQ for AIX Version 7.5
- IBM WebSphere MQ for HP-UX Version 7.5
- IBM WebSphere MQ for IBM i Version 7.5
- IBM WebSphere MQ for Linux Version 7.5
- IBM WebSphere MQ for Solaris Version 7.5
- IBM WebSphere MQ for Windows Version 7.5

Die Systemsteuerbefehle für jeweils einen Wert des Attributs *CommandLevel* hängen vom Wert des Attributs *Platform* ab; die Systemsteuerbefehle, die unterstützt werden, müssen über diese beiden Attribute festgelegt werden.

Der Wert dieses Attributs wird über den Selektor MQIA_COMMAND_LEVEL im Aufruf MQINQ ermittelt.

CommandServerControl (MQLONG)

Gibt an, ob der Befehlsserver beim Start des Warteschlangenmanagers gestartet werden soll.

Folgende Werte sind möglich:

MQSVC_CONTROL_MANUAL
Der Befehlsserver soll nicht automatisch gestartet werden.

MQSVC_CONTROL_Q_MGR
Der Befehlsserver soll beim Start des Warteschlangenmanagers automatisch gestartet werden.

Dieses Attribut wird unter z/OS nicht unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CMD_SERVER_CONTROL im MQINQ-Aufruf bestimmt.

ConfigurationEvent (MQLONG)

Steuert, ob Konfigurationsereignisse generiert werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CONFIGURATION_EVENT im MQINQ-Aufruf bestimmt.

Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

DeadLetterQName (MQCHAR48)

Dies ist der Name einer Warteschlange, die im lokalen Warteschlangenmanager als Warteschlange für nicht zustellbare Nachrichten definiert ist. An diese Warteschlange werden Nachrichten gesendet, die nicht an die korrekte Zieladresse weitergeleitet werden können.

Nachrichten werden zum Beispiel in folgenden Fällen in diese Warteschlange gestellt:

- In einem Warteschlangenmanager wird eine Nachricht für eine Warteschlange empfangen, die in dem Warteschlangenmanager noch nicht definiert ist.
- In einem Warteschlangenmanager wird eine Nachricht für eine Warteschlange empfangen, an die diese Nachricht möglicherweise aus den folgenden Gründen nicht weitergeleitet werden kann:
  - Die Warteschlange ist voll.
  - Put-Anforderungen werden unterdrückt.
  - Der sendende Knoten ist nicht berechtigt, Nachrichten in die Warteschlange einzureihen.

Auch Anwendungen können Nachrichten in die Warteschlange für nicht zustellbare Nachrichten einreihen.

Berichtsnachrichten werden genauso behandelt wie normale Nachrichten. Wenn die Berichtsnachricht nicht an die Zielwarteschlange übergeben werden kann (dies ist in der Regel die durch das Feld *ReplyToQ* im Nachrichtendeskriptor der ursprünglichen Nachricht angegebene Warteschlange), wird die Berichtsnachricht in der Warteschlange für unzustellbare Nachrichten abgelegt.

Anmerkung: Nachrichten, deren Ablaufzeit überschritten wurde (siehe MQMD - Expiry-Feld) werden beim Löschen **nicht** an diese Warteschlange übertragen. Es wird jedoch trotzdem eine Ablaufberichtsnachricht (MQRO_EXPIRATION) generiert und an die Warteschlange *ReplyToQ* gesendet, wenn die sendende Anwendung dies angefordert hat.

Nachrichten werden nicht in die Warteschlange für nicht zustellbare Nachrichten eingereiht, wenn die Anwendung, die die PUT-Anforderung ausgegeben hat, durch den vom MQPUT- oder MQPUT1-Aufruf zurückgegebenen Ursachencode synchron über das Problem benachrichtigt wurde (z. B. eine Nachricht, die in eine lokale Warteschlange eingereiht wurde, für die PUT-Anforderungen unterdrückt sind).

Den Anwendungsnachrichtendaten von Nachrichten in der Warteschlange für nicht zustellbare Nachrichten wird gelegentlich eine MQDLH-Struktur vorangestellt. Diese Struktur enthält Zusatzinformationen, die angeben, weshalb die Nachricht in der Warteschlange für nicht zustellbare Nachrichten platziert wurde. Weitere Informationen zu dieser Struktur finden Sie unter „MQDLH - Header für nicht zustellbare Nachrichten” auf Seite 329.

Diese Warteschlange muss eine lokale Warteschlange mit dem *Usage*-Attribut MQUS_NORMAL sein.

Wenn ein Warteschlangenmanager eine Warteschlange für nicht zustellbare Nachrichten nicht unterstützt oder keine definiert wurde, besteht der Name aus Leerzeichen. Alle WebSphere MQ-Warteschlangenmanager unterstützen eine Warteschlange für nicht zustellbare Nachrichten, in der Standardeinstellung ist diese jedoch nicht definiert.

Wenn die Warteschlange für nicht zustellbare Nachrichten nicht definiert, voll oder aus einem anderen Grund unbrauchbar ist, wird eine Nachricht, die über einen Nachrichtenkanalagent an die Warteschlange übertragen worden wäre, stattdessen in der Übertragungswarteschlange beibehalten.

Der Wert dieses Attributs wird mit dem Selektor MQCA_DEAD_LETTER_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

DefClusterXmitQueueType (MQLONG)

Das Attribut `DefClusterXmitQueueType` steuert, welche Übertragungswarteschlange standardmäßig von Clustersenderkanälen für den Abruf von Nachrichten ausgewählt wird, die an Clusterempfängerkanäle gesendet werden sollen.

Die Werte von `DefClusterXmitQueueType` sind `MQCLXQ_SCTQ` oder `MQCLXQ_CHANNEL`.

`MQCLXQ_SCTQ`

Alle Clustersenderkanäle senden Nachrichten von `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Die Korrelations-ID (`correlID`) der in die Übertragungswarteschlange gestellten Nachrichten gibt an, für welchen Clustersenderkanal die Nachricht bestimmt ist.

SCTQ wird bei der Definition eines Warteschlangenmanagers festgelegt. IBM WebSphere MQ-Versionen vor Version 7.5 weisen dieses Verhalten nur implizit auf. In früheren Versionen gab es das Warteschlangenmanagerattribut `DefClusterXmitQueueType` noch nicht.

`MQCLXQ_CHANNEL`

Jeder Clustersenderkanal sendet Nachrichten aus einer anderen Übertragungswarteschlange. Jede Übertragungswarteschlange wird als permanente dynamische Warteschlange aus der Modellwarteschlange `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE` erstellt.

Das Attribut wird unter z/OS nicht unterstützt.

Wenn das WS-Manager-Attribut `DefClusterXmitQueueType` als `CHANNEL` festgelegt wird, gilt Folgendes: wird die Standardkonfiguration in Clustersenderkanäle geändert, die einzelnen Clusterübertragungswarteschlangen zugeordnet sind. Die Übertragungswarteschlangen sind permanente dynamische Warteschlangen, die aus der Modellwarteschlange `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`erstellt werden. Jede Übertragungswarteschlange ist einem Clustersenderkanal zugeordnet. Da ein Clustersenderkanal eine Clusterübertragungswarteschlange bedient, enthält die Übertragungswarteschlange nur Nachrichten für einen einzigen Warteschlangenmanager in einem Cluster. Sie können Cluster so konfigurieren, dass jeder Warteschlangenmanager in einem Cluster nur eine einzige Clusterwarteschlange enthält. In diesem Fall erfolgt die Nachrichtenübertragung von einem Warteschlangenmanager an jede einzelne Clusterwarteschlange getrennt von Nachrichten an andere Warteschlangen.

Rufen Sie zum Abfragen des Werts `MQINQ` auf oder senden Sie einen PCF-Befehl 'Inquire Queue Manager' (`MQCMD_INQUIRE_Q_MGR`), der den Selektor `MQIA_DEF_CLUSTER_XMIT_Q_TYPE` festlegt. Senden Sie zum Ändern des Werts einen PCF-Befehl 'Change Queue Manager' (`MQCMD_CHANGE_Q_MGR`), der den Selektor `MQIA_DEF_CLUSTER_XMIT_Q_TYPE` festlegt.

Zugehörige Verweise

Warteschlangenmanager ändern
Warteschlangenmanager abfragen
„MQINQ - Objektattribute abfragen” auf Seite 700

Der Aufruf MQINQ gibt eine Ganzzahlenfeldgruppe und eine Zeichenfolgegruppe mit den Attributen eines Objekts zurück.

DefXmitQName (MQCHAR48)

Der Name der Übertragungswarteschlange, die für die Übertragung von Nachrichten an ferne Warteschlangenmanager verwendet wird, wenn keine weitere Angabe dazu vorhanden ist, welche Übertragungswarteschlange verwendet werden soll.

Wenn keine Standard-Übertragungs-WS vorhanden ist, bleibt der Name vollständig leer. Der Anfangswert dieses Attributs ist leer.

Der Wert dieses Attributs wird mit dem Selektor MQCA_DEF_XMIT_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

DistLists (MQLONG)

Das Attribut gibt an, ob der lokale Warteschlangenmanager mit den MQPUT- und MQPUT1-Aufrufen Verteilerlisten unterstützt. Folgende Werte sind möglich:

MQDL_SUPPORTED
Unterstützte Verteilerlisten.

MQDL_NOT_SUPPORTED
Nicht unterstützte Verteilerlisten.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DIST_LISTS im MQINQ-Aufruf bestimmt.

DNSGroup (MQCHAR18)

Der Name der Gruppe für das TCP-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet, wenn WLM/DNS-Unterstützung (Workload Manager für Dynamic Domain Name Services) verwendet wird. Die maximale Länge beträgt 18 Zeichen. Wenn Sie keinen Namen angeben, wird der Name der Gruppe mit gemeinsamer Warteschlange verwendet.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_DNS_GROUP im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_DNS_GROUP_NAME_LENGTH angegeben.

DNSWLM (MQLONG)

Gibt an, ob das TCP-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet, mit Workload Manager für dynamische Domänennamensservices registriert wird.

Folgende Werte sind möglich:

MQDNSWLM_YES
Das Empfangsprogramm wird für den Workload Manager registriert.

MQDNSWLM_NO
Das Empfangsprogramm wird nicht für den Workload Manager registriert. Dies ist der Standardwert.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DNS_WLM im MQINQ-Aufruf bestimmt.

ExpiryInterval (MQLONG)

Gibt die Häufigkeit an, mit der der Warteschlangenmanager Warteschlangen auf abgelaufene Nachrichten überprüft. Dies ist entweder ein Zeitintervall in Sekunden zwischen 1 und 99 999 999 oder der folgende Sonderwert:

MQEXPI_OFF
Der Warteschlangenmanager überprüft die Warteschlangen nicht auf abgelaufene Nachrichten.

Der Wert dieses Attributs wird mit dem Selektor MQIA_EXPIRY_INTERVAL im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

IGQPutAuthority (MQLONG)

Dieses Attribut gilt nur, wenn der lokale Warteschlangenmanager zu einer Gruppe mit gemeinsamer Warteschlange gehört. Es gibt die Art der Berechtigungsprüfung an, die ausgeführt wird, wenn der lokale Agent für die gruppeninterne Warteschlangensteuerung (IGQ-Agent) eine Nachricht aus der gemeinsamen Übertragungswarteschlange entfernt und in einer lokalen Warteschlange ablegt. Folgende Werte sind möglich:

MQIGQPA_DEFAULT
Die für die Autorisierung geprüfte Benutzer-ID ist der Wert des Feldes *UserIdentifier* in dem *separaten* Nachrichtendeskriptor, der der Nachricht zugeordnet ist, wenn sich die Nachricht in der gemeinsamen Übertragungswarteschlange befindet. Dies ist die Benutzer-ID des Programms, das die Nachricht in der gemeinsamen Übertragungswarteschlange abgelegt hat, und entspricht in der Regel der Benutzer-ID, mit der der ferne Warteschlangenmanager ausgeführt wird.

Wenn das Profil RESLEVEL angibt, dass mehrere Benutzer-IDs geprüft werden müssen, wird die Benutzer-ID des lokalen IGQ-Agenten (*IGQUserId*) ebenfalls geprüft.

MQIGQPA_CONTEXT
Die für die Autorisierung geprüfte Benutzer-ID ist der Wert des Feldes *UserIdentifier* in dem *separaten* Nachrichtendeskriptor, der der Nachricht zugeordnet ist, wenn sich die Nachricht in der gemeinsamen Übertragungswarteschlange befindet. Dies ist die Benutzer-ID des Programms, das die Nachricht in der gemeinsamen Übertragungswarteschlange abgelegt hat, und entspricht in der Regel der Benutzer-ID, mit der der ferne Warteschlangenmanager ausgeführt wird.

Wenn das Profil RESLEVEL angibt, dass mehrere Benutzer-IDs geprüft werden müssen, werden die Benutzer-ID des lokalen IGQ-Agenten (*IGQUserId*) und der Wert des Feldes *UserIdentifier* im *eingebetteten* Nachrichtendeskriptor ebenfalls geprüft. Bei der letzten Benutzer-ID handelt es sich in der Regel um die Benutzer-ID der Anwendung, von der die Nachricht stammt.

MQIGQPA_ONLY_IGQ
Die für die Autorisierung geprüfte Benutzer-ID ist die Benutzer-ID des lokalen IGQ-Agenten (*IGQUserId*).

Wenn das Profil RESLEVEL angibt, dass mehrere Benutzer-IDs geprüft werden müssen, wird diese Benutzer-ID für alle Prüfungen verwendet.

MQIGQPA_ALTERNATE_OR_IGQ
Die für die Autorisierung geprüfte Benutzer-ID ist die Benutzer-ID des lokalen IGQ-Agenten (*IGQUserId*).

Wenn das Profil RESLEVEL angibt, dass mehrere Benutzer-IDs geprüft werden müssen, wird der Wert des Feldes *UserIdentifier* im *eingebetteten* Nachrichtendeskriptor ebenfalls geprüft. Bei dieser Benutzer-ID handelt es sich in der Regel um die Benutzer-ID der Anwendung, von der die Nachricht stammt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_IGQ_PUT_AUTHORITY im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

IGQUserId (MQLONG)

Dieses Attribut gilt nur, wenn der lokale Warteschlangenmanager zu einer Gruppe mit gemeinsamer Warteschlange gehört. Es gibt die Benutzer-ID an, die dem lokalen Agenten für die gruppeninterne Warteschlangensteuerung (IGQ-Agent) zugeordnet ist. Diese ID ist eine der Benutzer-IDs, die für die Autorisierung geprüft werden können, wenn der IGQ-Agent Nachrichten in lokale Warteschlangen einreiht. Welche Benutzer-ID tatsächlich geprüft wird, hängt von der Einstellung des Attributs *IGQPutAuthority* und von externen Sicherheitsoptionen ab.

Wenn *IGQUserId* leer ist, ist dem IGQ-Agenten keine Benutzer-ID zugeordnet und die entsprechende Berechtigungsprüfung wird nicht ausgeführt (auch wenn andere Benutzer-IDs möglicherweise trotzdem für die Autorisierung geprüft werden).

Der Wert dieses Attributs wird mit dem Selektor MQCA_IGQ_USER_ID im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_USER_ID_LENGTH angegeben.

Dieses Attribut wird nur unter z/OS unterstützt.

InhibitEvent (MQLONG)

Steuert, ob Blockierungsereignisse (Sperren von GET- oder PUT-Operationen) erstellt werden. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_INHIBIT_EVENT im MQINQ-Aufruf bestimmt.

Unter z/OS kann der Wert dieses Attributs nicht mit dem MQINQ-Aufruf bestimmt werden.

IntraGroupQueuing (MQLONG)

Dieses Attribut gilt nur, wenn der lokale Warteschlangenmanager zu einer Gruppe mit gemeinsamer Warteschlange gehört. Es gibt an, ob die gruppeninterne Warteschlangensteuerung für die Gruppe mit gemeinsamer Warteschlange aktiviert ist. Folgende Werte sind möglich:

MQIGQ_DISABLED
Alle Nachrichten, die für andere Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange bestimmt sind, werden über herkömmliche Kanäle übertragen.

MQIGQ_ENABLED
Nachrichten, die für andere Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange bestimmt sind, werden über die gemeinsame Übertragungswarteschlange übertragen, wenn die folgende Bedingung erfüllt ist:

- Die Länge der Nachrichtendaten plus Übertragungsheader überschreitet nicht den Wert von 63 KB (64 512 Byte).

  Es wird empfohlen, etwas mehr Speicher als die Größe von MQXQH für den Übertragungsheader zuzuweisen. Zu diesem Zweck wird die Konstante MQ_MSG_HEADER_LENGTH bereitgestellt.

Wenn diese Bedingung nicht erfüllt ist, wird die Nachricht über herkömmliche Kanäle übertragen.

Anmerkung: Wenn die gruppeninterne Warteschlangensteuerung aktiviert ist, wird die Reihenfolge der über die gemeinsame Übertragungswarteschlange übertragenen Nachrichten im Verhältnis zu den über herkömmliche Kanäle übertragenen Nachrichten nicht beibehalten.

Der Wert dieses Attributs wird mit dem Selektor MQIA_INTRA_GROUP_QUEUING im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

IPAddressVersion (MQLONG)

Gibt an, welche IP-Adressenversion (IPv4 oder IPv6) verwendet wird.

Dieses Attribut ist nur für Systeme relevant, auf denen sowohl IPv4 als auch IPv6 ausgeführt wird und betrifft nur Kanäle, die gemäß Definition über einen *TransportType* von MQXPY_TCP verfügen, wenn eine der folgenden Bedingungen erfüllt ist:

- Der *ConnectionName* des Kanals ist ein Hostname, der sowohl in eine IPv4-Adresse als auch in eine IPv6-Adresse aufgelöst werden kann, und der Parameter *LocalAddress* ist nicht angegeben.

- Der *ConnectionName* und die *LocalAddress* des Kanals sind beide Hostnamen, die sowohl in IPv4-Adressen als auch in IPv6-Adressen aufgelöst werden können.

Folgende Werte sind möglich:

MQIPADDR_IPV4
IPv4 wird verwendet.

MQIPADDR_IPV6
IPv6 wird verwendet.

Der Wert dieses Attributs wird mit dem Selektor MQIA_IP_ADDRESS_VERSION im MQINQ-Aufruf ermittelt.

ListenerTimer (MQLONG)

Das Zeitintervall (in Sekunden) zwischen WebSphere MQ-Versuchen zum Neustart des Empfangsprogramms nach einem APPC- oder TCP/IP-Fehler. Der Wert muss zwischen 5 und 9999 liegen, wobei 60 der Standardwert ist.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_LISTENER_TIMER im MQINQ-Aufruf bestimmt.

LocalEvent (MQLONG)

Legt fest, ob lokale Fehlerereignisse generiert werden sollen. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_LOCAL_EVENT im MQINQ-Aufruf bestimmt.

Unter z/OS kann der Wert dieses Attributs nicht mit dem MQINQ-Aufruf bestimmt werden.

LoggerEvent (MQLONG)

Legt fest, ob Ereignisse für das Wiederherstellungsprotokoll generiert werden sollen. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_LOGGER_EVENT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris und Windowsunterstützt.

LUGroupName (MQCHAR8)

Der generische LU-Name für das LU 6.2-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet. Wenn Sie keinen Namen angeben, können Sie dieses Empfangsprogramm nicht verwenden.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_LU_GROUP_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_LU_NAME_LENGTH angegeben.

LUName (MQCHAR8)

Der Name der logischen Einheit, die für abgehende LU 6.2-Übertragungen verwendet werden soll. Legen Sie dieses Attribut auf dieselbe logische Einheit fest, die das Empfangsprogramm für eingehende Übertragungen verwendet. Wenn Sie keinen Namen angeben, wird die logische Standardeinheit APPC/MVS verwendet. Diese ist variabel, Sie sollten daher LUName immer festlegen, wenn Sie LU6.2 verwenden.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_LU_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_LU_NAME_LENGTH angegeben.

LU62ARMSuffix (MQCHAR2)

Das Suffix des SYS1.PARMLIB-Mitglieds APPCPMxx, das die LUADD für diesen Kanalinitiator benennt. Der z/OS-Befehl SET APPC=xx wird ausgegeben, wenn ARM den Kanalinitiator neu startet. Wenn Sie keinen Namen angeben, wird der Befehl SET APPC=xx nicht ausgegeben.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_LU62_ARM_SUFFIX im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_ARM_SUFFIX_LENGTH angegeben.

LU62Channels (MQLONG)

Die maximale Anzahl an aktiven Kanälen oder verbundenen Clients, die das Übertragungsprotokoll LU 6.2 verwenden.

Der Wert muss zwischen 0 und 9999 liegen, wobei 200 der Standardwert ist. Wenn Sie diesen Wert auf Null festlegen, wird das Übertragungsprotokoll LU 6.2 nicht verwendet.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_LU62_CHANNELS im MQINQ-Aufruf bestimmt.

MaxActiveChannels (MQLONG)

Dieses Attribut gibt die maximale Anzahl an Kanälen an, die jederzeit *aktiv* sein können.

Der Standardwert ist der Wert, der für das Attribut `MaxChannels` angegeben ist. Für z/OS muss der Wert zwischen 1 und 9.999 liegen. Für alle anderen Plattformen muss der Wert zwischen 1 und 65.535 liegen.

Der Wert dieses Attributs wird über den Selektor MQIA_ACTIVE_CHANNELS im Aufruf **MQINQ** ermittelt.

Zugehörige Konzepte

Kanalstatus

MaxChannels (MQLONG)

Dieses Attribut gibt die maximale Anzahl an Kanälen an, die *aktiv* sein können (einschließlich Serververbindungskanälen mit verbundenen Clients).

Für z/OS muss der Wert zwischen 1 und 9.999 liegen, wobei 200 der Standardwert ist. Für alle anderen Plattformen muss der Wert zwischen 1 und 65,535 liegen, wobei 100 der Standardwert ist. Ein System, das mit dem Bedienen von Verbindungen vom Netzwerk ausgelastet ist, benötigt unter Umständen eine höhere Zahl als die Standardeinstellung. Ermitteln Sie den korrekten Wert für Ihre Umgebung. Im Idealfall beobachten Sie dazu das Verhalten Ihres Systems während Tests.

Bei anderen Plattformen als z/OS wird der Wert für 'MaxChannels' in der Datei 'qm.ini' der betreffenden Warteschlangenmanager gesetzt.

Der Wert dieses Attributs wird über den Selektor MQIA_MAX_CHANNELS im Aufruf **MQINQ** ermittelt.

Zugehörige Konzepte

Kanalstatus

MaxHandles (MQLONG)

Die maximale Anzahl an Kennungen, die eine Aufgabe gleichzeitig verwenden kann. Jeder erfolgreiche MQOPEN-Aufruf für eine einzelne Warteschlange (oder für ein Objekt, das keine Warteschlange ist)

verwendet eine Kennung. Diese Kennung wird für die Wiederverwendung verfügbar, wenn das Objekt geschlossen wird. Wenn jedoch eine Verteilerliste geöffnet wird, wird jeder Warteschlange in der Verteilerliste eine separate Kennung zugewiesen, sodass der MQOPEN-Aufruf genauso viele Kennungen verwendet wie Warteschlangen in der Verteilerliste enthalten sind. Dies muss berücksichtigt werden, wenn über einen geeigneten Wert für *MaxHandles* entschieden wird.

Der MQPUT1-Aufruf führt einen MQOPEN-Aufruf als Teil seiner Verarbeitung durch; folglich verwendet MQPUT1 ebenso viele Kennungen wie MQOPEN, diese werden aber nur für die Dauer des MQPUT1-Aufrufs selbst verwendet.

Unter z/OS bezeichnet *Aufgabe* eine CICS-Aufgabe, eine MVS-Aufgabe oder eine IMS-abhängige Region.

Der Wert liegt in dem Bereich zwischen 0 und 999 999 999. Der Standardwert wird durch die Umgebung vorgegeben:

- Unter z/OS beträgt der Standardwert 100.
- In allen anderen Umgebungen beträgt der Standardwert 256.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_HANDLES im MQINQ-Aufruf bestimmt.

MaxMsgLength (MQLONG)

Die Länge der längsten *physischen* Nachricht, die ein Warteschlangenmanager verarbeiten kann. Da das Warteschlangenmanagerattribut *MaxMsgLength* jedoch unabhängig vom Warteschlangenattribut *MaxMsgLength* festgelegt werden kann, ist die längste physische Nachricht, die in eine Warteschlange eingereiht werden kann, der niedrigere dieser beiden Werte.

Wenn der Warteschlangenmanager die Segmentierung unterstützt, kann eine Anwendung eine *logische* Nachricht einreihen, die länger ist als der niedrigere Wert der beiden *MaxMsgLength*-Attribute. Dies ist jedoch nur möglich, wenn die Anwendung das Flag MQMF_SEGMENTATION_ALLOWED in MQMD angibt. Wenn dieses Flag angegeben ist, liegt die Obergrenze für die Länge einer logischen Nachricht bei 999 999 999 Byte. In der Regel führen jedoch vom Betriebssystem oder der Umgebung, in der die Anwendung ausgeführt wird, vorgegebene Ressourcenbeschränkungen zu einem niedrigeren Grenzwert.

Die Untergrenze für das Attribut *MaxMsgLength* liegt bei 32 KB (32 768 Byte). Die Obergrenze liegt bei 100 MB (104 857 600 Byte).

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_MSG_LENGTH im MQINQ-Aufruf bestimmt.

MaxPriority (MQLONG)

Dies ist der maximale Wert der Nachrichtenpriorität, der vom Warteschlangenmanager unterstützt wird. Die Prioritäten liegen zwischen null (am niedrigsten) und *MaxPriority* (am höchsten).

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_PRIORITY im MQINQ-Aufruf bestimmt.

MaxPropertiesLength (MQLONG)

Dieses Attribut wird verwendet, um die Größe der Eigenschaften zu steuern, die mit einer Nachricht übertragen werden können. Dies umfasst den Eigenschaftsnamen in Bytes und die Größe des Eigenschaftswerts in Bytes.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_PROPERTIES_LENGTH im MQINQ-Aufruf bestimmt.

MaxUncommittedMsgs (MQLONG)

Dies ist die maximale Anzahl nicht festgeschriebener Nachrichten, die innerhalb einer Arbeitseinheit vorhanden sein kann. Die Anzahl nicht festgeschriebener Nachrichten ist die Summe aus folgenden Elementen seit dem Start der aktuellen Arbeitseinheit:

- Von der Anwendung mit der Option MQPMO_SYNCPOINT eingereihte Nachrichten
- Von der Anwendung mit der Option MQGMO_SYNCPOINT abgerufene Nachrichten

- Auslösenachrichten und COA-Berichtsnachrichten, die vom Warteschlangenmanager für Nachrichten generiert werden, die mit der Option MQPMO_SYNCPOINT eingereiht wurden
- COD-Berichtsnachrichten, die vom Warteschlangenmanager für Nachrichten generiert wurden, die mit der Option MQGMO_SYNCPOINT abgerufen wurden

Folgende Nachrichten gelten *nicht* als nicht festgeschriebene Nachrichten:

- Nachrichten, die von der Anwendung außerhalb einer Arbeitseinheit eingereiht oder abgerufen werden
- Auslösenachrichten und COA-/COD-Berichtsnachrichten, die vom Warteschlangenmanager infolge von Nachrichten generiert werden, die außerhalb einer Arbeitseinheit eingereiht oder abgerufen werden
- Ablaufberichtsnachrichten, die vom Warteschlangenmanager generiert werden (auch wenn der Aufruf, der die Ablaufberichtsnachricht verursacht hat, MQGMO_SYNCPOINT angegeben hat)
- Ereignisnachrichten, die vom Warteschlangenmanager generiert werden (auch wenn der Aufruf, der die Ereignisnachricht verursacht hat, MQPMO_SYNCPOINT oder MQGMO_SYNCPOINT angegeben hat)

Anmerkung:

1. Abweichungsberichtsnachrichten werden vom Nachrichtenkanalagenten (MCA) oder von der Anwendung generiert und werden genauso behandelt wie normale Nachrichten, die von der Anwendung eingereiht oder abgerufen werden.
2. Wenn eine Nachricht oder ein Segment mit der Option MQPMO_SYNCPOINT eingereiht wird, wird die Anzahl nicht festgeschriebener Nachrichten unabhängig davon, wie viele physische Nachrichten tatsächlich aus der Einreihung resultieren, um Eins erhöht. (Wenn der Warteschlangenmanager die Nachricht oder das Segment unterteilen muss, werden unter Umständen mehrere physische Nachrichten generiert).
3. Wenn eine Verteilerliste mit der Option MQPMO_SYNCPOINT eingereiht wird, wird die Anzahl nicht festgeschriebener Nachrichten*für jede generierte physische Nachricht* um Eins erhöht. Diese Zahl kann gleich Eins oder der Anzahl der Ziele in der Verteilerliste sein.

Die Untergrenze für dieses Attribut ist 1; die Obergrenze ist 999 999 999. Der Standardwert ist 10000.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_UNCOMMITTED_MSGS im MQINQ-Aufruf bestimmt.

MQIAccounting (MQLONG)

Steuert die Erfassung von Abrechnungsinformationen für MQI-Daten.

Folgende Werte sind möglich:

MQMON_ON
API-Abrechnungsdaten werden erfasst.

MQMON_OFF
API-Abrechnungsdaten werden nicht erfasst. Dies ist der Standardwert.

Wenn Sie das Warteschlangenmanagerattribut ACCTCONO auf ENABLED setzen, wird dieser Wert unter Umständen für einzelne Verbindungen mit dem Options-Feld in der MQCNO-Struktur überschrieben. Änderungen an diesem Feld sind nur für Verbindungen zum Warteschlangenmanager wirksam, die nach der Änderung des Attributs hergestellt werden.

Dieses Attribut wird nur unter IBM i, UNIX-Systemen und Windows unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACCOUNTING_MQI im MQINQ-Aufruf bestimmt.

MQIStatistics (MQLONG)

Steuert die Erfassung von statischen Überwachungsdaten für den Warteschlangenmanager.

Folgende Werte sind möglich:

MQMON_ON
MQI-Statistikdaten werden erfasst.

MQMON_OFF
MQI-Statistikdaten werden nicht erfasst. Dies ist der Standardwert.

Dieses Attribut wird nur unter IBM i, UNIX and Linux-Systemen und Windows unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_STATISTICS_MQI im MQINQ-Aufruf bestimmt.

MsgMarkBrowseInterval (MQLONG)
Zeitintervall in Millisekunden, bevor der Warteschlangenmanager die Markierung automatisch vom Durchsuchen von Nachrichten löschen kann.

Dies ist ein Zeitintervall (in Millisekunden), bevor der Warteschlangenmanager die Markierung automatisch vom Durchsuchen von Nachrichten löschen kann.

Dieses Attribut legt das Zeitintervall fest, für das Nachrichten, die von einem MQGET-Aufruf über die Nachrichtenabrufoption MQGMO_MARK_BROWSE_CO_OP als durchsucht markiert wurden, als durchsucht markiert bleiben sollen.

Der Warteschlangenmanager kann automatisch die Markierung von durchsuchten Nachrichten, die für die mitwirkende Gruppe von Kennungen als durchsucht markiert wurden, aufheben, wenn sie nicht für länger als dieses ungefähre Intervall markiert wurden.

Dies betrifft nicht den Status von Nachrichten, die durch einen QGET-Abruf mit der Nachrichtenabrufoption MQGMO_MARK_BROWSE_HANDLE als durchsucht markiert wurden.

Der Maximalwert ist 999 999 999 und der Standardwert ist 5000. Der Sonderwert -1 für *MsgMarkBrowseInterval* stellt ein unbegrenztes Zeitintervall dar.

Achtung: Dieser Wert sollte nicht unter dem Standardwert 5000 liegen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MSG_MARK_BROWSE_INTERVAL im MQINQ-Aufruf bestimmt.

OutboundPortMax (MQLONG)
Die durch OutboundPortMin und OutboundPortMax definierte höchste Portnummer im Bereich der Portnummern, die zum Binden abgehender Kanäle verwendet werden sollen.

Der Wert ist eine ganze Zahl zwischen 0 und 65535 und muss größer oder gleich dem Wert für OutboundPortMin sein. Der Standardwert ist 0.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_OUTBOUND_PORT_MAX im MQINQ-Aufruf bestimmt.

OutboundPortMin (MQLONG)
Die durch OutboundPortMin und OutboundPortMax definierte niedrigste Portnummer im Bereich der Portnummern, die zum Binden abgehender Kanäle verwendet werden sollen.

Der Wert ist eine ganze Zahl zwischen 0 und 65535 und muss kleiner oder gleich dem Wert für OutboundPortMax sein. Der Standardwert ist 0.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_OUTBOUND_PORT_MIN im MQINQ-Aufruf bestimmt.

PerformanceEvent (MQLONG)

Legt fest, ob leistungsspezifische Ereignisse generiert werden. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_PERFORMANCE_EVENT im MQINQ-Aufruf bestimmt.

Plattform (MQLONG)

Gibt das Betriebssystem an, auf dem der Warteschlangenmanager ausgeführt wird:

MQPL_AIX
AIX (derselbe Wert wie MQPL_UNIX).

MQPL_MVS
z/OS (derselbe Wert wie MQPL_ZOS).

MQPL_NSK
HP Integrity NonStop Server.

MQPL_OS390
z/OS (derselbe Wert wie MQPL_ZOS).

MQPL_OS400
IBM i.

MQPL_UNIX
UNIX-Systeme.

MQPL_WINDOWS_NT
Windows-Systeme.

MQPL_ZOS
z/OS.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PLATFORM im MQINQ-Aufruf bestimmt.

PubSubNPInputMsg (MQLONG)

Gibt an, ob eine nicht zugestellte Eingabenachricht gelöscht oder aufbewahrt werden soll.

Folgende Werte sind möglich:

MQUNDELIVERED_DISCARD
Nicht persistente Eingabenachrichten können gelöscht werden, wenn sie nicht verarbeitet werden können.

Dies ist der Standardwert.

MQUNDELIVERED_KEEP
Nicht persistente Eingabenachrichten werden nicht gelöscht, wenn sie nicht verarbeitet werden. In dieser Situation versucht die Publish/Subscribe-Schnittstelle in der Warteschlange, den Prozess in angemessenen Intervallen zu wiederholen. Die Verarbeitung nachfolgender Nachrichten wird nicht fortgesetzt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PUBSUB_NP_MSG im MQINQ-Aufruf bestimmt.

PubSubNPResponse (MQLONG)

Steuert das Verhalten von nicht zugestellten Antwortnachrichten.

Folgende Werte sind möglich:

MQUNDELIVERED_NORMAL
Nicht persistente Antworten, die nicht in die Antwortwarteschlange eingereiht werden können, werden in die Warteschlange für nicht zustellbare Nachrichten eingereiht. Können sie nicht in die Warteschlange für nicht zustellbare Nachrichten eingereiht werden, dann werden sie gelöscht.

MQUNDELIVERED_SAFE
Nicht persistente Antworten, die nicht in die Antwortwarteschlange eingereiht werden können, werden in die Warteschlange für nicht zustellbare Nachrichten eingereiht. Wenn die Antwort nicht gesendet und nicht in der Warteschlange für nicht zustellbare Nachrichten eingereiht werden kann, dann führt die Publish/Subscribe-Schnittstelle in der Warteschlange für die aktuelle Operation eine

Rollback-Operation durch und wiederholt den Vorgang in angemessenen Intervallen. Die Verarbeitung nachfolgender Nachrichten wird nicht fortgesetzt.

MQUNDELIVERED_DISCARD
Nicht persistente Antworten werden nicht in die Antwortwarteschlange eingereiht und werden gelöscht.

Dies ist der Standardwert für neue Warteschlangenmanager.

MQUNDELIVERED_KEEP
Nicht persistente Antworten werden nicht in die Warteschlange für nicht zustellbare Nachrichten eingereiht und werden nicht gelöscht. Stattdessen setzt die Publish/Subscribe-Schnittstelle in der Warteschlange die aktuelle Operation zurück und wiederholt sie in angemessenen Intervallen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PUBSUB_NP_RESP im MQINQ-Aufruf bestimmt.

Standardwert für migrierte Warteschlangenmanager

Wenn der Warteschlangenmanager von WebSphere MQ V6.0 migriert wurde, dann ist der Anfangswert dieses Attributs von den Werten für DiscardNonPersistentResponse und DLQNonPersistentResponse abhängig, die vor der Migration definiert waren. Dieser Sachverhalt wird in der folgenden Tabelle dargestellt.

		DLQNonPersistentResponse		
		Ja	**Nein**	**Nicht festgelegt**
DiscardNonPersistentResponse	**Zulässig**	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD	MQUNDELIVERED_NORMAL
	Nicht zulässig	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	Nicht gesetzt	If SyncPointPersistent = No, MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL	If SyncPointPersistent = No, MQUNDELIVERED_KEEP else MQUNDELIVERED_DISCARD	If SyncPointPersistent = No, MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

Die Anzahl der Wiederholungen bei der Verarbeitung einer Nachricht zu einem fehlgeschlagenen Befehl unter dem Synchronisationspunkt.

Folgende Werte sind möglich:

0 - 999 999 999
Der Standardwert ist 5.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PUBSUB_MAXMSG_RETRY_COUNT im MQINQ-Aufruf bestimmt.

PubSubSyncPoint (MQLONG)

Gibt an, ob ausschließlich persistente Nachrichten oder aber alle Nachrichten unter dem Synchronisationspunkt verarbeitet werden sollen.

Folgende Werte sind möglich:

MQSYNCPOINT_IFPER
Dieser Wert führt dazu, dass die Publish/Subscribe-Schnittstelle in der Warteschlange nicht persistente Nachrichten außerhalb des Synchronisationspunkts empfängt. Wenn der Dämon eine Veröffentlichung außerhalb des Synchronisationspunkts empfängt, leitet er diese Veröffentlichung an ihm bekannte Subskribenten außerhalb des Synchronisationspunkts weiter.

Dies ist der Standardwert.

MQSYNCPOINT_YES
Dieser Wert führt dazu, dass die Publish/Subscribe-Schnittstelle in der Warteschlange alle Nachrichten unter dem Synchronisationspunkt empfängt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PUBSUB_SYNC_PT im MQINQ-Aufruf bestimmt.

PubSubMode (MQLONG)

Gibt an, ob die Publish/Subscribe-Engine und die Schnittstelle für eingereihtes Publish/Subscribe aktiv sind, sodass Anwendungen über die Anwendungsprogrammierschnittstelle und die Warteschlangen, die

von der Schnittstelle für eingereihtes Publish/Subscribe überwacht werden, Publish/Subscribe-Operationen durchführen können

Folgende Werte sind möglich:

MQPSM_COMPAT
Die Publish/Subscribe-Enging ist aktiv. Publish/Subscribe ist daher über die Anwendungsprogrammierschnittstelle möglich. Die eingereihte Publish/Subscribe-Schnittstelle ist nicht aktiv. Daher werden Nachrichten, die in die von der Schnittstelle für eingereihtes Publish/Subscribe überwachten Warteschlangen eingereiht werden, nicht verarbeitet. Diese Einstellung wird verwendet, um die Kompatibilität mit WebSphere Message Broker V6 oder früheren Versionen zu gewährleisten, die diesen Warteschlangenmanager verwenden, da er dieselben Warteschlangen lesen muss, die die eingereihte Publish/Subscribe-Schnittstelle normalerweise liest.

MQPSM_DISABLED
Die Publish/Subscribe-Engine und die Schnittstelle für eingereihtes Publish/Subscribe sind nicht aktiv. Publish/Subscribe ist daher über die Anwendungsprogrammierschnittstelle möglich. Publish/Subscribe-Nachrichten, die in die von der Schnittstelle für eingereihtes Publish/Subscribe überwachten Warteschlangen eingereiht werden, werden nicht verarbeitet.

MQPSM_ENABLED
Die Publish/Subscribe-Engine und die Schnittstelle für eingereihtes Publish/Subscribe sind aktiv. Daher ist Publish/Subscribe über die Anwendungsprogrammierschnittstelle und die Warteschlangen, die von der eingereihten Publish/Subscribe-Schnittstelle überwacht werden, möglich. Dies ist die anfängliche Standardeinstellung für den Warteschlangenmanager.

Der Wert dieses Attributs wird mit dem Selektor MQIA_PUBSUB_MODE im MQINQ-Aufruf bestimmt.

QMgrDesc (MQCHAR64)

Verwenden Sie dieses Feld für eine Erläuterung zur Beschreibung des Warteschlangenmanagers. Der Feldinhalt ist für den Warteschlangenmanager nicht von Bedeutung, es ist aber erforderlich, dass das Feld nur solche Zeichen enthält, die angezeigt werden können. Das Feld darf keine Nullzeichen enthalten; ggf. wird es rechts mit Leerzeichen aufgefüllt. In einer DBCS-Installation kann der Text DBCS-Zeichen enthalten (mit einer maximalen Feldlänge von 64 Byte).

Anmerkung: Wenn das Feld Zeichen enthält, die nicht Bestandteil des Zeichensatzes des Warteschlangenmanagers sind (gemäß der Definition im Warteschlangenmanagerattribut *CodedCharSetId*), werden sie möglicherweise falsch übersetzt, wenn das Feld an einen anderen Warteschlangenmanager gesendet wird.

- Unter z/OS ist der Standardwert der Produktname und die Versionsnummer.
- In allen anderen Umgebungen besteht der Standardwert aus Leerzeichen.

Der Wert dieses Attributs wird mit dem Selektor MQCA_Q_MGR_DESC im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_MGR_DESC_LENGTH angegeben.

QMgrIdentifier (MQCHAR48)

Ein intern generierter eindeutiger Name für den Warteschlangenmanager.

Der Wert dieses Attributs wird mit dem Selektor MQCA_Q_MGR_IDENTIFIER im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_MGR_IDENTIFIER_LENGTH angegeben.

Dieses Attribut wird in den folgenden Umgebungen unterstützt. AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windowsund WebSphere MQ -Clients, die mit diesen Systemen verbunden sind.

QMgrName (MQCHAR48)

Der Name des lokalen Warteschlangenmanagers, d. h. der Name des Warteschlangenmanagers, mit dem die Anwendung verbunden ist.

Die ersten zwölf Zeichen des Namens werden zum Erstellen einer eindeutigen Nachrichten-ID verwendet (siehe MQMD - MsgId-Feld). Warteschlangenmanager, die miteinander kommunizieren können, müssen

daher Namen aufweisen, die sich in den ersten zwölf Zeichen unterscheiden, damit Nachrichten-IDs im Warteschlangenmanagernetz eindeutig sind.

Unter z/OS entspricht der Name dem Subsystemnamen, der auf vier belegte Zeichen beschränkt ist.

Der Wert dieses Attributs wird mit dem Selektor MQCA_Q_MGR_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

QSGName (MQCHAR4)

Dies ist der Name der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört. Wenn der lokale Warteschlangenmanager keiner Gruppe mit gemeinsamer Warteschlange angehört, ist der Name leer.

Der Wert dieses Attributs wird mit dem Selektor MQCA_QSG_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_QSG_NAME_LENGTH angegeben.

Dieses Attribut wird nur unter z/OS unterstützt.

QueueAccounting (MQLONG)

Steuert die Erfassung von Abrechnungsinformationen für Warteschlangen.

Folgende Werte sind möglich:

MQMON_NONE
Unabhängig von der Einstellung des Warteschlangenabrechnungsattributs ACCTQ werden keine Abrechnungsdaten erfasst. Dies ist der Standardwert.

MQMON_OFF
Für Anforderungen, die QMGR im Warteschlangenattribut ACCTQ angeben, werden keine Abrechnungsdaten erfasst.

MQMON_ON
Für Warteschlangen, die QMGR im Warteschlangenattribut ACCTQ angeben, werden Abrechnungsdaten erfasst.

Änderungen an diesem Feld sind nur für Verbindungen zum Warteschlangenmanager wirksam, die nach der Änderung des Attributs hergestellt werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACCOUNTING_Q im Aufruf MQINQ bestimmt.

QueueMonitoring (MQLONG)

Gibt die Standardeinstellung für die Onlineüberwachung von Warteschlangen an.

Wenn das Warteschlangenattribut *QueueMonitoring* auf MQMON_Q_MGR gesetzt ist, gibt dieses Attribut den Wert an, der vom Kanal angenommen wird. Folgende Werte sind möglich:

MQMON_OFF
Die Erfassung von Onlineüberwachungsdaten ist inaktiviert. Dies ist die anfängliche Standardeinstellung für den Warteschlangenmanager.

MQMON_NONE
Die Erfassung von Onlineüberwachungsdaten für Warteschlangen wird unabhängig vom Wert des Attributs *QueueMonitoring* inaktiviert.

MQMON_LOW
Die Erfassung von Onlineüberwachungsdaten ist mit einer niedrigen Erfassungsrate aktiviert.

MQMON_MEDIUM
Die Erfassung von Onlineüberwachungsdaten ist mit einer mittleren Erfassungsrate aktiviert.

MQMON_HIGH
Die Erfassung von Onlineüberwachungsdaten ist mit einer hohen Erfassungsrate aktiviert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MONITORING_Q im MQINQ-Aufruf bestimmt.

QueueStatistics (MQLONG)

Steuert die Erfassung von Statistikdaten für Warteschlangen.

Folgende Werte sind möglich:

MQMON_NONE
Unabhängig von der Einstellung des Warteschlangenattributs *QueueStatistics* werden keine Statistikdaten für Warteschlangen erfasst. Dies ist der Standardwert.

MQMON_OFF
Für Warteschlangen, die Queue Manager im Warteschlangenattribut *QueueStatistics* angeben, werden keine Statistikdaten erfasst.

MQMON_ON
Für Warteschlangen, die Queue Manager im Warteschlangenattribut *QueueStatistics* angeben, werden Statistikdaten erfasst.

Der Wert dieses Attributs wird mit dem Selektor MQIA_STATISTICS_Q im MQINQ-Aufruf bestimmt.

ReceiveTimeout (MQLONG)
Gibt an, wie lange ein TCP/IP-Kanal ungefähr auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt. Das Attribut gilt nur für Nachrichtenkanäle und nicht für MQI-Kanäle.

Die genaue Bedeutung von ReceiveTimeout wird durch den in ReceiveTimeoutType angegebenen Wert geändert. ReceiveTimeoutType kann auf einen der folgenden Werte gesetzt werden:

- MQRCVTIME_EQUAL - dieser Wert ist die Wartezeit des Kanals in Sekunden. Geben Sie einen Wert zwischen 0 und 999999 an.
- MQRCVTIME_ADD - dieser Wert ist der Zeitraum in Sekunden, der dem verhandelten HBINT-Wert hinzugefügt wird, und legt fest, wie lange ein Kanal wartet. Geben Sie einen Wert zwischen 1 und 999999 an.
- MQRCVTIME_MULTIPLY - dieser Wert ist ein Multiplikator, der auf den verhandelten HBINT-Wert angewendet wird. Geben Sie den Wert 0 oder einen Wert zwischen 2 und 99 an.

Der Standardwert ist 0.

Legen Sie ReceiveTimeoutType auf MQRCVTIME_MULTIPLY oder MQRCVTIME_EQUAL und ReceiveTimeout auf 0 fest, um zu verhindern, dass die Wartezeit eines Kanals für den Empfang von Daten vom Partner abläuft.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_RECEIVE_TIMEOUT im MQINQ-Aufruf bestimmt.

ReceiveTimeoutMin (MQLONG)
Die Mindestzeitspanne in Sekunden, die ein TCP/IP-Kanal auf den Empfang von Daten, einschließlich Überwachungssignalen wartet, bevor er in den inaktiven Status zurückkehrt.

Das Attribut gilt nur für Nachrichtenkanäle, nicht für MQI-Kanäle. Der Wert muss zwischen 0 und 999999 liegen, wobei 0 der Standardwert ist.

Wenn Sie mit ReceiveTimeoutType angeben, dass die Wartezeit des TCP/IP-Kanals relativ zum verhandelten Wert von HBINT sein soll und der resultierende Wert kleiner ist als der Wert dieses Parameters, wird letzterer verwendet.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_RECEIVE_TIMEOUT_MIN im MQINQ-Aufruf bestimmt.

ReceiveTimeoutType (MQLONG)

Das auf ReceiveTimeout angewendete Qualifikationsmerkmal, um zu definieren, wie lange ein TCP/IP-Kanal auf den Empfang von Daten, einschließlich Überwachungssignalen, von seinem Partner wartet, bevor er in den inaktiven Status zurückkehrt. Das Attribut gilt nur für Nachrichtenkanäle, nicht für MQI-Kanäle.

Folgende Werte sind möglich:

MQRCVTIME_MULTIPLY
ReceiveTimeout ist ein Multiplikator, der auf den verhandelten HBINT-Wert angewendet wird, um zu ermitteln, wie lange ein Kanal wartet. Dies ist der Standardwert.

MQRCVTIME_ADD
ReceiveTimeout ist ein Wert in Sekunden, der dem verhandelten HBINT-Wert hinzugefügt wird, um zu ermitteln, wie lange ein Kanal wartet.

MQRCVTIME_EQUAL
ReceiveTimeout ist der Zeitraum in Sekunden, den ein Kanal wartet.

Um zu verhindern, dass die Wartezeit eines Kanals für den Empfang von Daten von seinem Partner abläuft, setzen Sie ReceiveTimeoutType auf MQRCVTIME_MULTIPLY oder MQRCVTIME_EQUAL und ReceiveTimeout auf 0.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_RECEIVE_TIMEOUT_TYPE im MQINQ-Aufruf bestimmt.

RemoteEvent (MQLONG)

Legt fest, ob ferne Fehlerereignisse generiert werden. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_REMOTE_EVENT im MQINQ-Aufruf bestimmt.

RepositoryName (MQCHAR48)

Dies ist der Name eines Clusters, für den dieser Warteschlangenmanager einen Repository-Manager-Service bereitstellt. Wenn der Warteschlangenmanager diesen Service mehr als einem Cluster zur Verfügung stellt, gibt *RepositoryNamelist* den Namen eines Namenslistenobjekts an, das die Cluster ermittelt, und für *RepositoryName* wird kein Wert angegeben. Mindestens einer der Werte für *RepositoryName* und *RepositoryNamelist* muss leer sein.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windowsund z/OSunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_REPOSITORY_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

RepositoryNamelist (MQCHAR48)

Dies ist der Name eines Namenslistenobjekts, das die Namen von Clustern enthält, für die dieser Warteschlangenmanager einen Repository-Service bereitstellt. Wenn die Warteschlange diesen Service nur für einen Cluster bereitstellt, enthält das Namenslistenobjekt nur einen Namen. Alternativ kann *RepositoryName* auch dazu verwendet werden, den Namen des Clusters anzugeben. In diesem Fall ist der Wert für *RepositoryNamelist* leer. Mindestens einer der Werte für *RepositoryName* und *RepositoryNamelist* muss leer sein.

Dieses Attribut wird nur unter AIX, HP-UX, IBM i, Linux, Solaris, Windowsund z/OSunterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_REPOSITORY_NAMELIST im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_NAMELIST_NAME_LENGTH angegeben.

ScyCase(MQCHAR8)

Gibt an, ob der Warteschlangenmanager Sicherheitsprofilnamen in Groß-/Kleinschreibung oder nur in Großschreibung unterstützt.

Folgende Werte sind möglich:

MQSCYC_UPPER
Sicherheitsprofilnamen müssen in Großbuchstaben angegeben werden.

MQSCYC_MIXED
Sicherheitsprofilnamen können in Großbuchstaben oder in Groß-/Kleinschreibung angegeben werden.

Änderungen an diesem Attribut werden wirksam, wenn ein Befehl zum Aktualisieren der Sicherheit ausgeführt wird, während *SecurityType(MQSECTYPE_CLASSES)* angegeben ist.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SECURITY_CASE im MQINQ-Aufruf bestimmt.

SharedQMgrName (MQLONG)

Gibt an, ob *ObjectQmgrName* als lokaler Warteschlangenmanager in einem MQOPEN-Aufruf für eine gemeinsame Warteschlange verwendet oder behandelt werden sollte, wenn der *ObjectQmgrName* der Name eines anderen Warteschlangenmanagers in der Gruppe mit gemeinsamer Warteschlange ist.

Folgende Werte sind möglich:

MQSQQM_USE
ObjectQmgrName wird verwendet und die entsprechende Übertragungswarteschlange geöffnet.

MQSQQM_IGNORE
Wenn die Zielwarteschlange gemeinsam genutzt wird und der *ObjectQmgrName* der Name eines Warteschlangenmanagers in derselben Gruppe mit gemeinsamer Warteschlange ist, erfolgt das Öffnen lokal.

Dieses Attribut ist nur unter z/OS gültig.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SHARED_Q_Q_MGR_NAME im 3MQINQ-Aufruf bestimmt.

SPLCAP

Gibt an, ob Sicherheitsfunktionen von WebSphere MQ Advanced Message Security für einen Warteschlangenmanager verfügbar sind.

MQCAP_SUPPORTED
Dies ist der Standardwert, wenn die Komponente WebSphere MQ AMS für die Installation installiert ist, in der der Warteschlangenmanager aktiv ist.

MQCAP_NOT_SUPPORTED

SSLEvent (MQLONG)

Gibt an, ob SSL-Ereignisse generiert werden.

Folgende Werte sind möglich:

MQEVR_ENABLED
SSL-Ereignisse werden wie folgt generiert:

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED
Es werden keine SSL-Ereignisse generiert. Dies ist der Standardwert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SSL_EVENT im MQINQ-Aufruf bestimmt.

SSLFIPSRequired (MQLONG)

Über dieses Attribut können Sie angeben, dass bei einer Ausführung der Verschlüsselung in WebSphere MQ anstatt in einer Verschlüsselungshardware nur FIPS-zertifizierte Algorithmen verwendet werden sollen. Wenn eine Verschlüsselungshardware konfiguriert ist, werden die vom Hardwareprodukt bereitgestellten Verschlüsselungsmodule verwendet; dabei kann es sich um (bis zu einem bestimmten FIPS-Level) FIPS-zertifizierte Module handeln, abhängig von der verwendeten Verschlüsselungshardware.

Folgende Werte sind möglich:

MQSSL_FIPS_NO
Es wird eine auf der jeweiligen Plattform unterstützte CipherSpec verwendet. Dies ist der Standardwert.

MQSSL_FIPS_YES
Es werden nur FIPS-zertifizierte Verschlüsselungsalgorithmen in den CipherSpecs verwendet, die für alle SSL-Verbindungen von und zu diesem Warteschlangenmanager zulässig sind.

Dieser Parameter ist nur auf UNIX-, Linux-, Windows-und z/OS -Plattformen gültig.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SSL_FIPS_REQUIRED im MQINQ-Aufruf bestimmt.

Zugehörige Tasks
Angeben, dass nur FIPS-zertifizierte CipherSpecs während der Ausführung auf dem MQI-Client verwendet werden
Zugehörige Verweise
Federal Information Processing Standards (FIPS) für UNIX, Linux und Windows

SSLKeyResetCount (MQLONG)
Gibt an, wann Nachrichtenkanalagenten (MCAs) im SSL-Kanal, die die Kommunikation initiieren, den zur Verschlüsselung des Kanals verwendeten geheimen Schlüssel zurücksetzen.

Der Wert stellt die Gesamtzahl unverschlüsselter Bytes dar, die vor einer Neuvereinbarung des geheimen Schlüssels im Kanal gesendet und empfangen werden. Die Anzahl der Byte enthält Steuerinformationen, die vom MCA gesendet wurden.

Der Wert ist eine Zahl im Bereich von 0 bis 999 999 999, wobei der Standardwert 0 ist. Wenn Sie für den Zählerstand für die Rücksetzung von geheimen SSL/TLS-Schlüsseln einen Wert zwischen 1 Byte und 32 KB setzen, verwenden die SSL/TLS-Kanäle als Zählerstand für die Rücksetzung des geheimen Schlüssels 32 KB. Dadurch wird der Aufwand für übermäßig viele Schlüsselrücksetzungen vermieden, wie es bei kleinen Rücksetzungswerten für geheime SSL/TSL-Schlüssel der Fall wäre.

Der geheime Schlüssel wird neu vereinbart, wenn die Gesamtzahl unverschlüsselter Bytes, die durch den Nachrichtenkanalagenten des initiierenden Kanals gesendet und empfangen werden, den angegebenen Wert überschreiten, oder wenn Kanalüberwachungssignale vor dem Senden von Daten aktiviert oder nach einem Kanalüberwachungssignal empfangen werden, je nachdem, welches Ereignis zuerst eintritt.

Die Anzahl der zur Neuvereinbarung gesendeten und empfangenen Bytes umfasst Steuerungsinformationen, die vom Nachrichtenkanalagenten des Kanals gesendet werden und bei jeder Neuvereinbarung zurückgesetzt werden.

Verwenden Sie den Wert 0, um anzugeben, dass geheime Schlüssel nie neu vereinbart werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SSL_RESET_COUNT im MQINQ-Aufruf bestimmt.

StartStopEvent (MQLONG)

Legt fest, ob Start- und Stoppereignisse generiert werden. Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_START_STOP_EVENT im MQINQ-Aufruf bestimmt.

StatisticsInterval (MQLONG)

Gibt an, wie oft (in Sekunden) statistische Überwachungsdaten in die Überwachungswarteschlange geschrieben werden sollen.

Der Wert ist eine Ganzzahl im Bereich von 0 bis 604800, mit einem Standardwert von 1800 (30 Minuten).

Der Wert dieses Attributs wird mit dem Selektor MQIA_STATISTICS_INTERVAL im MQINQ-Aufruf bestimmt.

SyncPoint (MQLONG)

Das Attribut gibt an, ob der lokale Warteschlangenmanager mit den Aufrufen MQGET, MQPUT und MQPUT1 Arbeitseinheiten und Synchronisationspunkte unterstützt.

MQSP_AVAILABLE
Arbeitseinheiten und Synchronisationspunkte verfügbar.

MQSP_NOT_AVAILABLE
Arbeitseinheiten und Synchronisationspunkte nicht verfügbar.

- Unter z/OS wird dieser Wert nie zurückgegeben.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SYNCPOINT im MQINQ-Aufruf bestimmt.

TCPChannels (MQLONG)

Die maximale Anzahl an aktiven Kanälen oder verbundenen Clients, die das TCP/IP-Übertragungsprotokoll verwenden.

Der Wert muss zwischen 0 und 9999 liegen, wobei 200 der Standardwert ist. Wenn Sie 0 angeben, wird TCP/IP nicht verwendet.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TCP_CHANNELS im MQINQ-Aufruf bestimmt.

TCPKeepAlive (MQLONG)

Gibt an, ob mithilfe von TCP KEEPALIVE überprüft werden soll, ob die Gegenseite der Verbindung noch verfügbar ist. Ist sie nicht verfügbar, wird der Kanal geschlossen.

Folgende Werte sind möglich:

MQTCPKEEP_YES
TCP KEEPALIVE wird verwendet, wie im Datensatz zur TCP-Profilkonfiguration angegeben. Wenn Sie das Kanalattribut KeepAliveInterval (KAINT) angeben, wird der Wert verwendet, auf den das Attribut gesetzt ist.

MQTCPKEEP_NO
TCP KEEPALIVE wird nicht verwendet. Dies ist der Standardwert.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TCP_KEEP_ALIVE im MQINQ-Aufruf bestimmt.

TCPName (MQCHAR8)

Der Name des einzigen oder standardmäßigen TCP/IP-Systems, das Sie verwenden (abhängig vom Wert für TCPStackType). Der Standardwert ist TCPIP.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_TCP_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_TCP_NAME_LENGTH angegeben.

TCPStackType (MQLONG)

Gibt an, ob der Kanalinitiator nur den in TCPName angegebenen TCP/IP-Adressraum verwenden oder optional eine Bindung zu einer beliebigen ausgewählten TCP/IP-Adresse herstellen kann.

Folgende Werte sind möglich:

MQTCPSTACK_SINGLE
Der Kanalinitiator darf nur den TCP/IP-Adressraum verwenden, der in TCPName angegeben wurde. Dies ist der Standardwert.

MQTCPSTACK_MULTIPLE
Der Kanalinitiator kann jeden beliebigen verfügbaren TCP/IP-Adressraum verwenden. Wird für einen Kanal oder ein Empfangsprogramm kein bestimmter Adressraum angegeben, wird standardmäßig der in TCPName angegebene Adressraum verwendet.

Dieses Attribut wird nur unter z/OS unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TCP_STACK_TYPE im MQINQ-Aufruf bestimmt.

TraceRouteRecording (MQLONG)

Steuert die Aufzeichnung von Trace-Route-Daten.

Folgende Werte sind möglich:

MQRECORDING_DISABLED
Anhängen an Trace-Route-Nachrichten nicht zulässig.

MQRECORDING_Q
Trace-Route-Nachrichten werden in eine Warteschlange mit festgelegtem Namen eingereiht.

MQRECORDING_MSG
Trace-Route-Nachrichten werden in eine Warteschlange eingereiht, die mithilfe der Nachricht selbst festgelegt wird. Dies ist der Standardwert

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRACE_ROUTE_RECORDING im MQINQ-Aufruf bestimmt.

TriggerInterval (MQLONG)

Mit diesem Zeitintervall (Angabe in Millisekunden) wird die Anzahl der Auslösenachrichten beschränkt. Das Attribut ist nur dann relevant, wenn für *TriggerType* die Option MQTT_FIRST angegeben wurde. In diesem Fall werden Auslösenachrichten normalerweise nur dann erstellt, wenn eine entsprechende Nachricht in der Warteschlange eingeht und diese zuvor leer war. Unter bestimmten Umständen kann jedoch eine weitere Auslösenachricht (mit MQTT_FIRST als Auslösetyp) generiert werden, auch wenn die Warteschlange nicht leer war. Diese zusätzlichen Auslösenachrichten werden in einem Zeitabstand erstellt, der durch das Attribut *TriggerInterval* in Millisekunden angegeben wird.

Weitere Informationen zum Auslösen finden Sie unter Kanäle auslösen .

Der Wert ist nicht kleiner als 0 und nicht größer als 999 999 999. Der Standardwert ist 999 999 999.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_INTERVAL im MQINQ-Aufruf bestimmt.

TriggerInterval (MQLONG)

Mit diesem Zeitintervall (Angabe in Millisekunden) wird die Anzahl der Auslösenachrichten beschränkt. Das Attribut ist nur dann relevant, wenn für *TriggerType* die Option MQTT_FIRST angegeben wurde. In diesem Fall werden Auslösenachrichten normalerweise nur dann erstellt, wenn eine entsprechende Nachricht in der Warteschlange eingeht und diese zuvor leer war. Unter bestimmten Umständen kann jedoch eine weitere Auslösenachricht (mit MQTT_FIRST als Auslösetyp) generiert werden, auch wenn die Warteschlange nicht leer war. Diese zusätzlichen Auslösenachrichten werden in einem Zeitabstand erstellt, der durch das Attribut *TriggerInterval* in Millisekunden angegeben wird.

Weitere Informationen zum Auslösen finden Sie unter Kanäle auslösen .

Der Wert ist nicht kleiner als 0 und nicht größer als 999 999 999. Der Standardwert ist 999 999 999.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_INTERVAL im MQINQ-Aufruf bestimmt.

Version (MQCFST)

Die Version des WebSphere MQ-Codes als VVRRMMFF, wobei:

VV - Version

RR - Release

MM - Wartungsstufe

FF - Fixversion

XrCapability(MQLONG)

Dieses Attribut gibt an, ob der Warteschlangenmanager WebSphere MQ Telemetry-Befehle unterstützt.

Folgende Werte sind möglich:

MQCAP_SUPPORTED
Die Komponente WebSphere MQ Telemetry ist installiert und Telemetry-Befehle werden unterstützt.

MQCAP_NOT_SUPPORTED
WebSphere MQ Telemetry ist nicht installiert.

Dieses Attribut wird nur auf IBM i-, UNIX- und Windows-Systemen unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_XR_CAPABILITY im Aufruf MQINQ bestimmt.

Attribute für Warteschlangen

Es gibt fünf Warteschlangendefinitionstypen. Einige Warteschlangenattribute gelten für alle Warteschlangentypen, andere nur für bestimmte Warteschlangentypen.

Warteschlangentypen

Der Warteschlangenmanager unterstützt folgende Warteschlangendefinitionstypen:

Lokale Warteschlange
Sie können Nachrichten in einer lokalen Warteschlange speichern. Unter z/OS können Sie sie als gemeinsam genutzte oder private Warteschlange festlegen.

Eine Warteschlange gilt für ein Programm als *lokal*, wenn sie dem Warteschlangenmanager zugeordnet ist, mit dem das Programm verbunden ist. Nachrichten können aus lokalen Warteschlangen abgerufen und darin eingereiht werden.

Das Warteschlangendefinitionsobjekt enthält die Definitionsinformationen der Warteschlange sowie die in die Warteschlange eingereihten physischen Nachrichten.

Warteschlange des lokalen Warteschlangenmanagers
Die Warteschlange befindet sich im lokalen Warteschlangenmanager. Diese Warteschlange wird unter z/OS private Warteschlange genannt.

Gemeinsam genutzte Warteschlange (nur z/OS)
Die Warteschlange befindet sich in einem gemeinsam genutzten Repository, auf das alle Warteschlangenmanager zugreifen können, die der Gruppe mit gemeinsamer Warteschlange angehören, die Eigner des Repositorys ist.

Anwendungen, die mit einem der Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange verbunden sind, können Nachrichten in Warteschlangen dieses Typs stellen und Nachrichten daraus entfernen. Solche Warteschlangen funktionieren auf dieselbe Weise wie lokale Warteschlangen. Der Wert des Warteschlangenattributs *QType* ist MQQT_LOCAL.

Anwendungen, die mit dem lokalen Warteschlangenmanager verbunden sind, können Nachrichten in Warteschlangen dieses Typs stellen und Nachrichten daraus entfernen. Der Wert des Warteschlangenattributs *QType* ist MQQT_LOCAL.

Clusterwarteschlange

Sie können Nachrichten in einer Clusterwarteschlange in dem Warteschlangenmanager speichern, bei dem sie definiert ist. Eine Clusterwarteschlange wird von einem Clusterwarteschlangenmanager anderen Warteschlangenmanagern im Cluster zur Verfügung gestellt. Der Wert des Warteschlangenattributs *QType* ist MQQT_CLUSTER.

Eine Clusterwarteschlangendefinition wird den anderen Warteschlangenmanagern im Cluster zugänglich gemacht. Die anderen Warteschlangenmanager im Cluster können ohne entsprechende Definition einer fernen Warteschlange Nachrichten in eine Clusterwarteschlange einreihen. Über eine Clusternamensliste kann eine Clusterwarteschlange in mehreren Clustern zugänglich gemacht werden.

Wenn eine Warteschlange zugänglich gemacht wird, können alle Warteschlangenmanager im Cluster Nachrichten in diese Warteschlange einreihen. Um eine Nachricht einzureihen, muss der Warteschlangenmanager anhand der vollständigen Repositorys ermitteln, wo sich die Warteschlange befindet. Anschließend fügt der Warteschlangenmanager der Nachricht einige Routing-Informationen hinzu und stellt sie dann in eine Clusterübertragungswarteschlange.

Abgesehen von z/OS kann ein Warteschlangenmanager Nachrichten für andere Warteschlangenmanager in einem Cluster in mehrere Übertragungswarteschlangen stellen. Es gibt zwei Möglichkeiten, einen Warteschlangenmanager so zu konfigurieren, dass er Nachrichten in mehreren Clusterübertragungswarteschlangen speichern kann. Wenn Sie das Warteschlangenmanagerattribut DEFCLXQ auf CHANNEL setzen, wird für jeden Clustersenderkanal automatisch eine andere Clusterübertragungswarteschlange anhand von SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE erstellt. Wenn Sie die Option CLCHNAME für eine Clusterübertragungswarteschlange so setzen, dass sie mit einem oder auch mehreren Clustersenderkanälen übereinstimmt, kann der Warteschlangenmanager in dieser Übertragungswarteschlange Nachrichten für diese Clustersenderkanäle speichern.

Eine Clusterwarteschlange kann eine Warteschlange sein, die von Mitgliedern einer Gruppe mit gemeinsamer Warteschlange in IBM WebSphere MQ for z/OS gemeinsam genutzt wird.

Ferne Warteschlange

Eine ferne Warteschlange ist keine physische Warteschlange, sondern die lokale Definition einer Warteschlange, die sich in einem fernen Warteschlangenmanager befindet. Die lokale Definition der fernen Warteschlange enthält Informationen, die dem lokalen Warteschlangenmanager mitteilen, wie er Nachrichten an den fernen Warteschlangenmanager weiterleiten kann.

Anwendungen, die mit dem lokalen Warteschlangenmanager verbunden sind, können Nachrichten in Warteschlangen dieses Typs stellen. Die Nachrichten werden in die lokale Übertragungswarteschlange gestellt, die zur Weiterleitung von Nachrichten an den fernen Warteschlangenmanager verwendet wird. Anwendungen können keine Nachrichten aus fernen Warteschlangen entfernen. Der Wert des Warteschlangenattributs *QType* ist MQQT_REMOTE.

Die Definition einer fernen Warteschlange kann auch für folgende Zwecke verwendet werden:

- Aliasnamensumsetzung für Antwortwarteschlange

  In diesem Fall ist der Name der Definition der Name der Empfangswarteschlange für Antworten. Weitere Informationen hierzu finden Sie unter Cluster und Aliasnamen für Warteschlangen für Antwortnachrichten.

- Aliasnamensumsetzung für Warteschlangenmanager

  In diesem Fall ist der Name der Definition ein Aliasname für einen Warteschlangenmanager und nicht der Name einer Warteschlange. Weitere Informationen hierzu finden Sie unter Cluster und Aliasnamen für Warteschlangenmanager.

Aliaswarteschlange

Dies ist keine physische Warteschlange, sondern ein alternativer Name für eine lokale Warteschlange, eine gemeinsam genutzte Warteschlange, eine Clusterwarteschlange oder eine ferne Warteschlange. Der Name der Warteschlange, in den der Aliasname aufgelöst wird, ist Teil der Definition der Aliaswarteschlange.

Anwendungen, die mit dem lokalen Warteschlangenmanager verbunden sind, können Nachrichten in Warteschlangen dieses Typs stellen. Die Nachrichten werden in die Warteschlange gestellt, in

deren Name der Aliasname aufgelöst wird. Anwendungen können Nachrichten aus Warteschlangen dieses Typs entfernen, wenn der Aliasname in eine lokale Warteschlange, eine gemeinsam genutzte Warteschlange oder eine Clusterwarteschlange mit einer lokalen Instanz aufgelöst wird. Der Wert des Warteschlangenattributs *QType* ist MQQT_ALIAS.

Modellwarteschlange
Dies ist keine physische Warteschlange, sondern eine Gruppe von Warteschlangenattributen, aus denen eine lokale Warteschlange erstellt werden kann.

In Warteschlangen dieses Typs können keine Nachrichten gespeichert werden.

Warteschlangenattribute

Einige Warteschlangenattribute gelten für alle Warteschlangentypen, andere nur für bestimmte Warteschlangentypen. Die Warteschlangentypen, für die ein Attribut gilt, sind in Tabelle 573 auf Seite 835 und den nachfolgenden Tabellen aufgeführt.

Tabelle 573 auf Seite 835 enthält eine Zusammenfassung der Attribute, die für Warteschlangen spezifisch sind. Die Attribute werden in alphabetischer Reihenfolge beschrieben.

Anmerkung: Die Namen der in diesem Abschnitt gezeigten Attribute sind beschreibende Namen, die mit den Aufrufen MQINQ und MQSET verwendet werden; es sind dieselben Namen wie für die PCF-Befehle. Wenn Attribute mit MQSC-Befehlen definiert, geändert oder angezeigt werden, werden alternative Kurznamen verwendet. Weitere Informationen finden Sie unter Scriptbefehle (MQSC).

Tabelle 573. Attribute für Warteschlangen. Bedeutung der Spalten:

- Die Spalte für lokale Warteschlangen ist auch für gemeinsam genutzte Warteschlangen gültig.
- Die Spalte für Modellwarteschlangen gibt an, welche Attribute von der lokalen Warteschlange, die aus der Modellwarteschlange erstellt wird, übernommen werden.
- Die Spalte für Clusterwarteschlangen gibt die Attribute an, die abgefragt werden können, wenn die Clusterwarteschlange nur für Abfragen oder für Abfragen und Ausgaben geöffnet wird. Wenn andere Attribute abgefragt werden, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück.

  Wenn die Clusterwarteschlange für Abfragen und zusätzlich für eine oder mehrere Eingabe-, Anzeige- oder Einreihungsoperationen geöffnet wird, gilt stattdessen die Spalte für lokale Warteschlangen.

  Wenn die Clusterwarteschlange für die Abfrage allein oder für die Abfrage und Ausgabe geöffnet wird und der Name des Basiswarteschlangenmanagers angegeben wird, gilt stattdessen die Spalte für lokale Warteschlangen.

Attribut	Beschreibung	Lokal	Modell	Alias	Fern	Cluster
AlterationDate	Datum der letzten Änderung der Definition	✓		✓	✓	
AlterationTime	Uhrzeit der letzten Änderung der Definition	✓		✓	✓	
BackoutRequeueQName	Name der Warteschlange zum Wiedereinreihen überzähliger zurückgesetzter Nachrichten	✓	✓			
BackoutThreshold	Rücksetzschwellenwert	✓	✓			
BaseQName	Warteschlangenname, in den der Aliasname aufgelöst wird			✓		
CFStrucName	Name der Coupling-Facility-Struktur	✓	✓			
CLCHNAME	Clustersenderkanalnamen	✓	✓			
ClusterName	Name des Clusters, zu dem die Warteschlange gehört	✓		✓	✓	✓
ClusterNamelist	Name des Namenslistenobjekts mit den Namen von Clustern, zu denen die Warteschlange gehört	✓		✓	✓	
CLWLQueuePriority	Warteschlangenpriorität für Clusterauslastung	✓		✓	✓	✓
CLWLQueueRank	Warteschlangenrangfolge für Clusterauslastung	✓		✓	✓	✓

Tabelle 573. Attribute für Warteschlangen. Bedeutung der Spalten:

- Die Spalte für lokale Warteschlangen ist auch für gemeinsam genutzte Warteschlangen gültig.
- Die Spalte für Modellwarteschlangen gibt an, welche Attribute von der lokalen Warteschlange, die aus der Modellwarteschlange erstellt wird, übernommen werden.
- Die Spalte für Clusterwarteschlangen gibt die Attribute an, die abgefragt werden können, wenn die Clusterwarteschlange nur für Abfragen oder für Abfragen und Ausgaben geöffnet wird. Wenn andere Attribute abgefragt werden, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück.

  Wenn die Clusterwarteschlange für Abfragen und zusätzlich für eine oder mehrere Eingabe-, Anzeige- oder Einreihungsoperationen geöffnet wird, gilt stattdessen die Spalte für lokale Warteschlangen.

  Wenn die Clusterwarteschlange für die Abfrage allein oder für die Abfrage und Ausgabe geöffnet wird und der Name des Basiswarteschlangenmanagers angegeben wird, gilt stattdessen die Spalte für lokale Warteschlangen.

(Forts.)

Attribut	Beschreibung	Lokal	Modell	Alias	Fern	Cluster
CLWLUseQ	Ferne Warteschlange verwenden	✓				
CreationDate	Erstellungsdatum der Warteschlange	✓				
CreationTime	Erstellungszeit der Warteschlange	✓				
CurrentQDepth	Aktuelle Warteschlangenlänge	✓				
DefaultPutResponse	Standard-PUT-Antwort	✓	✓	✓	✓	
DefBind	Standardbindung	✓		✓	✓	✓
DefinitionType attribute	Warteschlangendefinitionstyp	✓	✓			
DefInputOpenOption	Standardoption für die Öffnung zur Eingabe	✓	✓			
DefPersistence	Standardpersistenz für Nachrichten	✓	✓	✓	✓	✓
DefPriority	Standardpriorität für Nachr.	✓	✓	✓	✓	✓
DefReadAhead	Standardvorauslesen	✓	✓	✓		
DistLists	Unterstützung Verteilerliste	✓	✓			
HardenGetBackout	Gibt an, ob ein genauer Rücksetzungszähler verwaltet werden soll	✓	✓			
IndexType	Indextyp	✓	✓			
InhibitGet	Gibt an, ob GET-Operationen für die Warteschlange zulässig sind	✓	✓	✓		
InhibitPut	Gibt an, ob PUT-Operationen für die Warteschlange zulässig sind	✓	✓	✓	✓	✓
InitiationQName	Name der Initialisierungswarteschlange	✓	✓			
MaxMsgLength	Maximale Nachrichtenlänge in Byte	✓	✓			
MaxQDepth	Maximale Warteschlangenlänge	✓	✓			
MsgDeliverySequence attribute	Reihenfolge bei der Nachrichtenübertragung	✓	✓			
NonPersistentMessage Class	Zuverlässigkeitsziel für nicht persistente Nachrichten	✓	✓			
OpenInputCount	Anzahl der Operationen zum Öffnen für Eingaben	✓				
OpenOutputCount	Anzahl der Operationen zum Öffnen für Ausgaben	✓				

Tabelle 573. Attribute für Warteschlangen. Bedeutung der Spalten:

- Die Spalte für lokale Warteschlangen ist auch für gemeinsam genutzte Warteschlangen gültig.
- Die Spalte für Modellwarteschlangen gibt an, welche Attribute von der lokalen Warteschlange, die aus der Modellwarteschlange erstellt wird, übernommen werden.
- Die Spalte für Clusterwarteschlangen gibt die Attribute an, die abgefragt werden können, wenn die Clusterwarteschlange nur für Abfragen oder für Abfragen und Ausgaben geöffnet wird. Wenn andere Attribute abgefragt werden, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück.

  Wenn die Clusterwarteschlange für Abfragen und zusätzlich für eine oder mehrere Eingabe-, Anzeige- oder Einreihungsoperationen geöffnet wird, gilt stattdessen die Spalte für lokale Warteschlangen.

  Wenn die Clusterwarteschlange für die Abfrage allein oder für die Abfrage und Ausgabe geöffnet wird und der Name des Basiswarteschlangenmanagers angegeben wird, gilt stattdessen die Spalte für lokale Warteschlangen.

(Forts.)

Attribut	Beschreibung	Lokal	Modell	Alias	Fern	Cluster
PropertyControl	Eigenschaftensteuerung	✓	✓	✓		
ProcessName	Prozessname	✓	✓			
QDepthHighEvent attribute	Gibt an, ob 'Warteschlangenlänge hoch'-Ereignisse generiert werden	✓	✓			
QDepthHighLimit	Oberer Grenzwert für Warteschlangenlänge	✓	✓			
QDepthLowEvent attribute	Gibt an, ob 'Warteschlangenlänge niedrig'-Ereignisse generiert werden	✓	✓			
QDepthLowLimit attribute	Unterer Grenzwert für Warteschlangenlänge	✓	✓			
QDepthMaxEvent	Gibt an, ob 'Warteschlange voll'-Ereignisse generiert werden	✓	✓			
QDesc	Warteschlangenbeschreibung	✓	✓	✓	✓	✓
QName	Warteschlangenname	✓		✓	✓	✓
QServiceInterval	Ziel für Warteschlangenserviceintervall	✓	✓			
QServiceIntervalEvent attribute	Gibt an, ob 'Serviceintervall hoch'- oder 'Serviceintervall OK'-Ereignisse generiert werden	✓	✓			
QSGDisp attribute	Disposition der Gruppe mit gemeinsamer Warteschlange	✓		✓	✓	
QueueAccounting	Erfassung von WS-Abrechnungsdaten	✓	✓	✓	✓	✓
QueueMonitoring	Onlineüberwachungsdaten für Warteschlangen	✓	✓			
QueueStatistics	Erfassung statistischer Warteschlangendaten	✓	✓	✓	✓	✓
QType	Warteschlangentyp	✓		✓	✓	✓
RemoteQMgrName	Name des fernen Warteschlangenmanagers				✓	
RemoteQName	Name der fernen Warteschlange				✓	
RetentionInterval	Rückhalteintervall	✓	✓			
Scope	Gibt an, ob ein Eintrag für die Warteschlange auch in einem Zellenverzeichnis steht	✓		✓	✓	
Shareability	Gemeinsame Nutzung der Warteschlange	✓	✓			
StorageClass	Speicherklasse für Warteschlange	✓	✓			

Tabelle 573. Attribute für Warteschlangen. Bedeutung der Spalten:

- Die Spalte für lokale Warteschlangen ist auch für gemeinsam genutzte Warteschlangen gültig.
- Die Spalte für Modellwarteschlangen gibt an, welche Attribute von der lokalen Warteschlange, die aus der Modellwarteschlange erstellt wird, übernommen werden.
- Die Spalte für Clusterwarteschlangen gibt die Attribute an, die abgefragt werden können, wenn die Clusterwarteschlange nur für Abfragen oder für Abfragen und Ausgaben geöffnet wird. Wenn andere Attribute abgefragt werden, gibt der Aufruf den Beendigungscode MQCC_WARNING und den Ursachencode MQRC_SELECTOR_NOT_FOR_TYPE (2068) zurück.

  Wenn die Clusterwarteschlange für Abfragen und zusätzlich für eine oder mehrere Eingabe-, Anzeige- oder Einreihungsoperationen geöffnet wird, gilt stattdessen die Spalte für lokale Warteschlangen.

  Wenn die Clusterwarteschlange für die Abfrage allein oder für die Abfrage und Ausgabe geöffnet wird und der Name des Basiswarteschlangenmanagers angegeben wird, gilt stattdessen die Spalte für lokale Warteschlangen.

(Forts.)

Attribut	Beschreibung	Lokal	Modell	Alias	Fern	Cluster
TriggerControl	Auslösesteuerung	✓	✓			
TriggerData	Daten des Auslösers	✓	✓			
TriggerDepth	Auslösertiefe	✓	✓			
TriggerMsgPriority	Schwellenwertnachrichtenpriorität für Auslöser	✓	✓			
TriggerType	Auslösertyp	✓	✓			
Usage attribute	Warteschlangennutzung	✓	✓			
XmitQName	Name der Übertragungswarteschlange				✓	

Zugehörige Konzepte
Clusterwarteschlangen
Lokale Warteschlangen

AlterationDate (MQCHAR12)

Datum der letzten Änderung der Definition.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	

Dieses Attribut gibt das Datum an, an dem die Definition zuletzt geändert wurde. Das Format des Datums ist YYYY-MM-DD, das mit zwei abschließenden Leerzeichen aufgefüllt wird, damit die Länge 12 Byte beträgt (z. B. 1992-09-23¬¬, wobei ¬¬ zwei Leerzeichen darstellt).

Die Werte bestimmter Attribute (z. B. *CurrentQDepth*) ändern sich, während der Warteschlangenmanager ausgeführt wird. Änderungen an diesen Attributen haben keine Auswirkungen auf *AlterationDate*.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_DATE im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_DATE_LENGTH vorgegeben.

AlterationTime (MQCHAR8)

Uhrzeit der letzten Änderung der Definition.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	

Dieses Attribut gibt die Uhrzeit an, zu der die Definition zuletzt geändert wurde. Das Zeitformat lautet HH.MM.SS und wird im 24-Stunden-Format angegeben. Wenn die Stunde kleiner als 10 ist, wird eine führende Null hinzugefügt (z. B. 09.10.20).

- Unter z/OS ist die Uhrzeit Greenwich Mean Time (GMT), sofern die Systemuhr präzise auf GMT eingestellt ist.
- In anderen Umgebungen entspricht die Uhrzeit der Ortszeit.

Die Werte bestimmter Attribute (z. B. *CurrentQDepth*) ändern sich, während der Warteschlangenmanager ausgeführt wird. Änderungen an diesen Attributen haben keine Auswirkung auf *AlterationTime*.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_TIME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_TIME_LENGTH vorgegeben.

BackoutRequeueQName (MQCHAR48)

Dies ist der Name der Warteschlange zum Wiedereinreihen überzähliger zurückgesetzter Nachrichten. Der Wert dieses Attributs kann abgefragt werden, Aktionen des Warteschlangenmanagers auf Basis dieses Wertes erfolgen nicht.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Anwendungen, die in WebSphere Application Server ausgeführt werden oder die WebSphere MQ Application Server Facilities nutzen, verwenden dieses Attribut um festzulegen, wohin Nachrichten übergeben werden sollen, die zurückgesetzt wurden. Bei allen anderen Anwendungen erfolgt keine Aktion des Warteschlangenmanagers auf Basis des Werts dieses Attributs.

Die WebSphere MQ-Klassen für Java Message Service verwenden dieses Attribut, um zu bestimmen, wohin eine Nachricht übertragen wird, die bereits die im Attribut *BackoutThreshold* angegeben maximale Anzahl zurückgesetzt wurde.

Der Wert dieses Attributs wird mit dem Selektor MQCA_BACKOUT_REQ_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

BackoutThreshold (MQLONG)

Dies ist der Rücksetzschwellenwert. Der Wert dieses Attributs kann abgefragt werden, Aktionen des Warteschlangenmanagers auf Basis dieses Wertes erfolgen nicht.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Anwendungen, die in WebSphere Application Server ausgeführt werden oerr die WebSphere MQ Application Server Facilities nutzen, verwenden dieses Attribut um festzulegen, ob eine Nachricht zurückgesetzt werden soll. Bei allen anderen Anwendungen erfolgt keine Aktion des Warteschlangenmanagers auf Basis des Werts dieses Attributs.

Die WebSphere MQ-Klassen für JMS verwenden dieses Attribut, um zu bestimmen, wie oft eine Nachricht zurückgesetzt werden kann, bevor sie an die im Attribut *BackoutRequeueQName* angegebene Warteschlange weitergeleitet wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_BACKOUT_THRESHOLD im Aufruf MQINQ bestimmt.

BaseQName (MQCHAR48)

Dies ist der Name einer Warteschlange, die für den lokalen Warteschlangenmanager definiert ist.

Lokal	Modell	Alias	Fern	Cluster
		X		

(Weitere Informationen zu Warteschlangen finden Sie im Abschnitt MQOD - ObjectName-Feld.) Folgende Typen sind für die Warteschlange zulässig:

MQQT_LOCAL
Lokale Warteschlange.

MQQT_REMOTE
Lokale Definition einer fernen Warteschlange.

MQQT_CLUSTER
Clusterwarteschlange.

Der Wert dieses Attributs wird mit dem Selektor MQCA_BASE_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

BaseType (MQCFIN)
Der Objekttyp, in den der Aliasname aufgelöst wird.

Lokal	Modell	Alias	Fern	Cluster
		X		

Folgende Werte sind möglich:

MQOT_Q
Basisobjekttyp ist eine Warteschlange.

MQOT_TOPIC
Der Basisobjekttyp ist ein Thema.

CFStrucName (MQCHAR12)
Dies ist der Name der Coupling-Facility-Struktur, in der die Nachrichten der Warteschlange gespeichert werden. Das erste Zeichen des Namens befindet sich im Bereich A bis Z, und die übrigen Zeichen sind im Bereich A bis Z, 0 bis 9 oder leer.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Um den vollständigen Namen der Struktur in der Coupling-Facility abzurufen, fügen Sie an den Wert des Warteschlangenmanagerattributs *QSGName* den Wert des Warteschlangenattributs *CFStrucName* an.

Dieses Attribut gilt nur für gemeinsam genutzte Warteschlangen, es wird ignoriert, wenn *QSGDisp* nicht den Wert MQQSGD_SHARED hat.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CF_STRUC_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_CF_STRUC_NAME_LENGTH angegeben.

Dieses Attribut wird nur unter z/OS unterstützt.

ClusterChannelName (MQCHAR20)
`ClusterChannelName` ist der generische Name der Clustersenderkanäle, die diese Warteschlange als Übertragungswarteschlange verwenden. Das Attribut gibt an, über welche Clustersenderkanäle Nachrichten aus dieser Clusterübertragungswarteschlange an einen Clusterempfängerkanal gesendet werden. `ClusterChannelName` wird unter z/OS nicht unterstützt.

Lokal	Modell	Alias	Fern	Cluster
✓	✓			

Laut Standardkonfiguration für Warteschlangenmanager senden alle Clustersenderkanäle Nachrichten aus einer einzigen Übertragungswarteschlange (SYSTEM.CLUSTER.TRANSMIT.QUEUE). Die Standardkonfiguration kann geändert werden, indem das Warteschlangenmanagerattribut DefClusterXmitQueueType geändert wird. Der Standardwert des Attributs ist SCTQ. Sie können diesen Wert in CHANNEL ändern. Wenn Sie das Attribut DefClusterXmitQueueTyp auf CHANNELsetzen, verwendet jeder Clustersenderkanal standardmäßig eine bestimmte Clusterübertragungswarteschlange, SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

Sie können das Attribut ClusterChannelName der Übertragungswarteschlange auch manuell auf einen Clustersenderkanal setzen. Nachrichten, die für einen Warteschlangenmanager bestimmt sind, der über einen Clustersenderkanal verbunden ist, werden in der Übertragungswarteschlange gespeichert, die den Clustersenderkanal angibt. Sie werden nicht in der standardmäßigen Clusterübertragungswarteschlange gespeichert. Wenn Sie für das Attribut ClusterChannelName Leerzeichen angeben, schaltet der Kanal bei einem Neustart auf die standardmäßige Clusterübertragungswarteschlange um. Die Standardwarteschlange ist entweder SYSTEM.CLUSTER.TRANSMIT.*ChannelName* oder SYSTEM.CLUSTER.TRANSMIT.QUEUE, abhängig vom Wert des Warteschlangenmanagerattributs DefClusterXmitQueueType.

Durch die Angabe von Sternen ("*") in ClusterChannelName können Sie eine Übertragungswarteschlange einer Gruppe von Clustersenderkanälen zuordnen. Die Sterne können am Anfang, am Ende oder auch an jeder Stelle in der Zeichenfolge mit dem Kanalnamen angegeben werden. ClusterChannelName ist auf eine Länge von 20 Zeichen begrenzt: MQ_CHANNEL_NAME_LENGTH.

ClusterName (MQCHAR48)

Dies ist der Name des Clusters, zu dem die Warteschlange gehört.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Wenn die Warteschlange zu mehr als einem Cluster gehört, gibt *ClusterNamelist* den Namen eines Namenslistenobjekts an, das die Cluster ermittelt, und für *ClusterName* erfolgt keine Angabe. Mindestens eines der Attribute *ClusterName* und *ClusterNamelist* muss leer sein.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CLUSTER_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_CLUSTER_NAME_LENGTH angegeben.

ClusterNamelist (MQCHAR48)

Hierbei handelt es sich um den Namen eines Namenslistenobjekts, das die Namen von Clustern enthält, zu denen diese Warteschlange gehört.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	

Wenn die Warteschlange nur zu einem Cluster gehört, enthält das Namenslistenobjekt nur einen Namen. Alternativ kann *ClusterName* auch dazu verwendet werden, den Namen des Clusters anzugeben. In diesem Fall ist der Wert für *ClusterNamelist* leer. Mindestens eines der Attribute *ClusterName* und *ClusterNamelist* muss leer sein.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CLUSTER_NAMELIST im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_NAMELIST_NAME_LENGTH angegeben.

CLWLQueuePriority (MQLONG)

Dies ist die Warteschlangenpriorität für Clusterauslastung, ein Wert im Bereich von 0 bis 9, der die Priorität der Warteschlange angibt.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Weitere Informationen finden Sie unter Clusterwarteschlangen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CLWL_Q_PRIORITY im MQINQ-Aufruf bestimmt.

CLWLQueueRank (MQLONG)

Dies ist der Warteschlangenrangordnung für Clusterauslastung, ein Wert im Bereich von 0 bis 9, der den Rang der Warteschlange angibt.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Weitere Informationen finden Sie unter Clusterwarteschlangen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CLWL_Q_RANK im MQINQ-Aufruf bestimmt.

CLWLUseQ (MQLONG)

Dies Attribut definiert das Verhalten eines MQPUT-Aufrufs, wenn die Zielwarteschlange sowohl eine lokale Instanz als auch mindestens eine ferne Clusterinstanz hat. Wenn das Einreihen aus einem Clusterkanal stammt, gilt dieses Attribut nicht.

Lokal	Modell	Alias	Fern	Cluster
X				

Folgende Werte sind möglich:

MQCLWL_USEQ_ANY
Ferne und lokale Warteschlangen verwenden

MQCLWL_USEQ_LOCAL
Es werden keine fernen Warteschlangen verwendet.

MQCLWL_USEQ_AS_Q_MGR
Definition von MQIA_CLWL_USEQ des Warteschlangenmanagers übernehmen

Weitere Informationen finden Sie unter Clusterwarteschlangen.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CLWL_USEQ im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_CLWL_USEQ_LENGTH angegeben.

CreationDate (MQCHAR12)

Dies ist das Datum, an dem die Warteschlange erstellt wurde.

Lokal	Modell	Alias	Fern	Cluster
X				

The format of the date is `YYYY-MM-DD`, padded with two trailing blanks to make the length 12 bytes (for example, `2013-09-23¬¬`, where ¬¬ represents 2 blank characters).

- Unter IBM i kann sich das Erstellungsdatum einer Warteschlange von dem der zugrundeliegenden Betriebssystementität (Datei oder Benutzeradressbereich) unterscheiden, die die Warteschlange darstellt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CREATION_DATE im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_CREATION_DATE_LENGTH angegeben.

CreationTime (MQCHAR8)

Dies ist der Zeitpunkt, zu dem die Warteschlange erstellt wurde.

Lokal	Modell	Alias	Fern	Cluster
X				

Das Zeitformat lautet `HH.MM.SS` und wird im 24-Stunden-Format angegeben. Wenn die Stunde kleiner als 10 ist, wird eine führende Null hinzugefügt (z. B. `09.10.20`).

- Unter z/OS ist die Uhrzeit Greenwich Mean Time (GMT), sofern die Systemuhr präzise auf GMT eingestellt ist.
- In anderen Umgebungen entspricht die Uhrzeit der Ortszeit.
- Unter IBM i kann sich die Erstellungszeit einer Warteschlange von der der zugrundeliegenden Betriebssystementität (Datei oder Benutzeradressbereich) unterscheiden, die die Warteschlange darstellt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_CREATION_TIME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_CREATION_TIME_LENGTH angegeben.

CurrentQDepth (MQLONG)

Dies ist die Anzahl der Nachrichten, die sich momentan in der Warteschlange befinden.

Lokal	Modell	Alias	Fern	Cluster
X				

Der Wert des Attributs wird während des MQPUT-Aufrufs und während des Zurücksetzens des MQGET-Aufrufs erhöht. Er wird während des MQGET-Aufrufs (nicht Anzeige) und während des Zurücksetzens des MQPUT-Aufrufs verringert. Dies bewirkt, dass der Zähler Nachrichten umfasst, die in die Warteschlange in einer Arbeitseinheit eingereiht, aber noch nicht festgeschrieben wurden, auch wenn sie nicht mit dem MQGET-Aufruf abgerufen werden können. Gleichermaßen werden Nachrichten ausgeschlossen, die in einer Arbeitseinheit mit dem MQGET-Aufruf abgerufen, aber noch nicht festgeschrieben wurden.

Der Zähler umfasst auch Nachrichten, die ihre Ablaufzeit überschritten haben, aber noch nicht verworfen wurden, auch wenn sie nicht abgerufen werden können. Weitere Informationen finden Sie im Abschnitt MQMD - Feld mit Ablaufinformation.

Sowohl Arbeitseinheitenverarbeitung als auch Segmentierung von Nachrichten kann bewirken, dass *CurrentQDepth* den Wert von *MaxQDepth* überschreitet. Dies wirkt sich allerdings nicht auf die Abrufbarkeit der Nachrichten aus; *alle* Nachrichten in der Warteschlange können auf normale Art mit dem MQGET-Aufruf abgerufen werden.

Der Wert dieses Attributs ändert sich, während der Warteschlangenmanager ausgeführt wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_CURRENT_Q_DEPTH im MQINQ-Aufruf bestimmt.

DefaultPutResponse (MQLONG)

Gibt den Typ der Antwort an, die für PUT-Operationen an die Warteschlange verwendet wird, wenn eine Anwendung MQPMO_RESPONSE_AS_Q_DEF angibt.

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	

Folgende Werte sind möglich:

MQPRT_SYNC_RESPONSE
Die PUT-Operation wird synchron ausgegeben und gibt eine Antwort zurück.

MQPRT_ASYNC_RESPONSE
Die Put-Operation wird asynchron ausgegeben und gibt eine Untermenge von MQMD-Feldern zurück.

DefBind (MQLONG)

Dies ist die Standardbindung, die verwendet wird, wenn MQOO_BIND_AS_Q_DEF für den MQOPEN-Aufruf angegeben wird und die Warteschlange eine Clusterwarteschlange ist.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Folgende Werte sind möglich:

MQBND_BIND_ON_OPEN
Bindung durch MQOPEN-Aufruf festgelegt.

MQBND_BIND_NOT_FIXED
Bindung nicht festgelegt.

MQBND_BIND_ON_GROUP
Mit dieser Option kann eine Anwendung fordern, dass alle Nachrichten einer Nachrichtengruppe an dieselbe Zielinstanz übergeben werden. Da dieser Wert in IBM WebSphere MQ Version 7.1 neu ist, darf er nicht verwendet werden, wenn Anwendungen, die diese Warteschlange öffnen, eine Verbindung zu IBM WebSphere MQ Version 7.0.1 oder früheren Warteschlangenmanagern herstellen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEF_BIND im MQINQ-Aufruf bestimmt.

DefinitionType (MQLONG)
Gibt an, wie die Warteschlange definiert wurde.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Folgende Werte sind möglich:

MQQDT_PREDEFINED
Die Warteschlange ist eine permanente, vom Systemadministrator erstellte Warteschlange; nur der Systemadministrator kann sie löschen.

Vordefinierte Warteschlangen werden mithilfe des MQSC-Befehls DEFINE erstellt und können nur mithilfe des MQSC-Befehls DELETE wieder gelöscht werden. Vordefinierte Warteschlangen können nicht auf der Basis von Modellwarteschlangen erstellt werden.

Befehle können entweder von einem Operator oder einem berechtigten Benutzer ausgegeben werden, der eine Befehlsnachricht an die Befehlseingabewarteschlange sendet (weitere Informationen finden Sie im Abschnitt CommandInputQName-Attribut).

MQQDT_PERMANENT_DYNAMIC
Die Warteschlange ist eine permanente Warteschlange, die von einer Anwendung erstellt wurde, die einen MQOPEN-Aufruf mit dem Namen einer Modellwarteschlange im Objektdeskriptor MQOD ausgibt. Der Wert des Attributs *DefinitionType* der Modellwarteschlangendefinition war MQQDT_PERMANENT_DYNAMIC.

Dieser Warteschlangentyp kann mit dem MQCLOSE-Aufruf gelöscht werden. Weitere Informationen finden Sie in „MQCLOSE - Objekt schließen” auf Seite 642.

Der Wert des Attributs *QSGDisp* für eine permanente dynamische Warteschlange ist MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC
Die Warteschlange ist eine temporäre Warteschlange, die von einer Anwendung erstellt wurde, die den MQOPEN-Aufruf mit dem Namen einer im Objektdeskriptor MQOD angegebenen Modellwarteschlange ausgegeben hat. Der Wert des Attributs *DefinitionType* der Modellwarteschlangendefinition war MQQDT_TEMPORARY_DYNAMIC.

Dieser Warteschlangentyp wird vom MQCLOSE-Aufruf automatisch gelöscht, wenn er von der Anwendung, die ihn erstellt hat, geschlossen wird.

Der Wert des Attributs *QSGDisp* für eine temporäre dynamische Warteschlange ist MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC
Die Warteschlange ist eine gemeinsam genutzte permanente Warteschlange, die von einer Anwendung erstellt wurde, die einen MQOPEN-Aufruf mit dem Namen einer Modellwarteschlange im Objektdeskriptor MQOD ausgibt. Der Wert des Attributs *DefinitionType* der Modellwarteschlangendefinition war MQQDT_SHARED_DYNAMIC.

Dieser Warteschlangentyp kann mit dem MQCLOSE-Aufruf gelöscht werden. Weitere Informationen finden Sie in „MQCLOSE - Objekt schließen” auf Seite 642.

Der Wert des Attributs *QSGDisp* für eine gemeinsam genutzte dynamische Warteschlange ist MQQSGD_SHARED.

Dieses Attribut gibt in der Definition einer Modellwarteschlange nicht an, auf welche Weise die Modellwarteschlange definiert wurde, da Modellwarteschlangen immer vordefiniert sind. Stattdessen wird der Wert dieses Attributs verwendet, um den *DefinitionType* jeder dynamischen Warteschlange zu bestimmen, die aus der Modellwarteschlangendefinition mit dem MQOPEN-Aufruf erstellt wurde.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEFINITION_TYPE im MQINQ-Aufruf bestimmt.

DefInputOpenOption (MQLONG)

Dies ist die Standardeinstellung, in der die Warteschlange für die Eingabe geöffnet werden soll.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Das Attribut wird angewendet, wenn die Option MQOO_INPUT_AS_Q_DEF im MQOPEN-Aufruf angegeben ist, wenn die Warteschlange geöffnet wird. Folgende Werte sind möglich:

MQOO_INPUT_EXCLUSIVE
Öffnet eine Warteschlange zum Abrufen von Nachrichten mit exklusivem Zugriff.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen geöffnet. Der Aufruf schlägt mit Ursachencode MQRC_OBJECT_IN_USE fehl, wenn die Warteschlange zuvor von dieser oder einer anderen Anwendung für eine beliebige Art der Eingabe (MQOO_INPUT_SHARED oder MQOO_INPUT_EXCLUSIVE) geöffnet wurde.

MQOO_INPUT_SHARED
Öffnet eine Warteschlange zum Abrufen von Nachrichten mit gemeinsamem Zugriff.

Die Warteschlange wird zur Verwendung mit nachfolgenden MQGET-Aufrufen geöffnet. Der Aufruf kann erfolgreich ausgeführt werden, wenn die Warteschlange zuvor von dieser oder einer anderen Anwendung mit MQOO_INPUT_SHARED geöffnet wurde, schlägt jedoch mit Ursachencode MQRC_OBJECT_IN_USE fehl, wenn die Warteschlange zuvor mit MQOO_INPUT_EXCLUSIVE geöffnet wurde.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEF_INPUT_OPEN_OPTION im MQINQ-Aufruf verwendet.

DefPersistence (MQLONG)

Dies ist die Standardpersistenz von Nachrichten in der Warteschlange. Dies gilt, wenn MQPER_PERSISTENCE_AS_Q_DEF im Nachrichtendeskriptor angegeben wird, wenn die Nachricht eingereiht wird.

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	X

Wenn im Auflösungspfad des Warteschlangennamens mehr als eine Definition vorhanden ist, wird zum Zeitpunkt des MQPUT- oder MQPUT1-Aufrufs die Standardpersistenz dem Wert dieses Attributs in der *ersten* im Pfad angegebenen Definition entnommen. Dies können sein:

- Eine Aliaswarteschlange
- Eine lokale Warteschlange
- Eine lokale Definition einer fernen Warteschlange
- Ein Warteschlangenmanageralias
- Eine Übertragungswarteschlange (z. B. die Warteschlange *DefXmitQName*)

Folgende Werte sind möglich:

MQPER_PERSISTENT
Die Nachricht bleibt bei Systemausfällen und Warteschlangenmanagerneustarts erhalten. Persistente Nachrichten können in folgenden Warteschlangen nicht platziert werden:

- Temporäre dynamische Warteschlangen
- Gemeinsam genutzte Warteschlangen, die einem CFSTRUCT-Objekt auf CFLEVEL(2) oder darunter zugeordnet sind, oder bei denen das CFSTRUCT-Objekt als RECOVER(NO) definiert ist.

Persistente Nachrichten können in permanente dynamische Warteschlangen und in vordefinierte Warteschlangen eingefügt werden.

MQPER_NOT_PERSISTENT
Die Nachricht bleibt bei Systemausfällen und Warteschlangenmanagerneustarts nicht erhalten. Dies gilt auch, wenn eine intakte Kopie der Nachricht bei einem Warteschlangenmanagerneustart in einem Zusatzspeicher gefunden wird.

Bei gemeinsam genutzten Warteschlangen bleiben nicht persistente Nachrichten bei einem Warteschlangenmanagerneustart in der Gruppe mit gemeinsamer Warteschlange *erhalten*, aber nicht bei Ausfall der Coupling-Facility, die verwendet wird, um Nachrichten in den gemeinsam genutzten Warteschlangen zu speichern.

Sowohl persistente als auch nicht persistente Nachrichten können in derselben Warteschlange vorhanden sein.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEF_PERSISTENCE im MQINQ-Aufruf bestimmt.

DefPriority (MQLONG)
Dies ist die Standardpriorität für Nachrichten in der Warteschlange. Dies gilt, wenn MQPRI_PRIORITY_AS_Q_DEF im Nachrichtendeskriptor angegeben wird, wenn die Nachricht in die Warteschlange eingereiht wird.

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	X

Wenn im Auflösungspfad des Warteschlangennamens mehr als eine Definition vorhanden ist, wird zum Zeitpunkt der Put-Operation die Standardpriorität für die Nachricht dem Wert dieses Attributs in der *ersten* im Pfad angegebenen Definition entnommen. Dies können sein:

- Eine Aliaswarteschlange
- Eine lokale Warteschlange
- Eine lokale Definition einer fernen Warteschlange
- Ein Warteschlangenmanageralias
- Eine Übertragungswarteschlange (z. B. die Warteschlange *DefXmitQName*)

Die Art und Weise, wie eine Nachricht in einer Warteschlange platziert wird, hängt von dem Wert des Attributs *MsgDeliverySequence* der Warteschlange ab.

- Wenn das Attribut *MsgDeliverySequence* den Wert MQMDS_PRIORITY hat, wird die logische Position, an der eine Nachricht in die Warteschlange eingefügt wird, vom Wert des Felds *Priority* im Nachrichtendeskriptor bestimmt.
- Wenn das Attribut *MsgDeliverySequence* den Wert MQMDS_FIFO hat, werden Nachrichten entsprechend der Priorität *DefPriority* der aufgelösten Warteschlange in die Warteschlange eingefügt, unabhängig vom Wert des Feldes *Priority* im Nachrichtendeskriptor. Das Feld *Priority* behält allerdings den Wert bei, den die Anwendung angegeben hat, die die Nachricht eingereiht hat. Weitere Informationen hierzu finden Sie im Abschnitt MsgDeliverySequence-Attribut.

Prioritäten liegen im Bereich null (Minimum) bis *MaxPriority* (Maximum), siehe MaxPriority-Attribut.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEF_PRIORITY im MQINQ-Aufruf bestimmt.

DefReadAhead (MQLONG)

Gibt das standardmäßige Vorausleseverhalten für nicht persistente Nachrichten an den Client an.

Lokal	Modell	Alias	Fern	Cluster
X	X	X		

DefReadAhead kann auf einen der folgenden Werte gesetzt werden:

MQREADA_NO
Nicht persistente Nachrichten werden nicht vorab an den Client gesendet, bevor sie von einer Anwendung angefordert werden. Bei abnormaler Beendigung des Clients kann maximal eine nicht persistente Nachricht verloren gehen.

MQREADA_YES
Nicht persistente Nachrichten werden an den Client vorausgesendet, bevor eine Anwendung sie anfordert. Nicht persistente Nachrichten können verloren gehen, wenn der Client abnormal endet oder wenn der Client nicht alle Nachrichten, die ihm gesendet werden, liest.

MQREADA_DISABLED
Für diese Warteschlange ist das Vorauslesen nicht persistenter Nachrichten nicht aktiviert. Nachrichten werden nicht an den Client gesendet, unabhängig davon, ob Vorauslesen von der Clientanwendung angefordert ist.

Der Wert dieses Attributs wird über den Selektor MQIA_DEF_READ_AHEAD im MQINQ-Aufruf ermittelt.

DefPResp (MQLONG)

Das Standardattribut für den PUT-Antworttyp (DEFPRESP) definiert den Wert, der von Anwendungen verwendet wird, wenn der PutResponseType in MQPMO auf MQPMO_RESPONSE_AS_Q_DEF gesetzt ist. Dieses Attribut ist für alle Warteschlangentypen gültig.

Lokal	Modell	Alias	Fern	Cluster
Ja	Ja	Ja	Ja	Ja

Folgende Werte sind möglich:

SYNC
Die Put-Operation wird synchron ausgegeben und gibt eine Antwort zurück.

ASYNC
Die Put-Operation wird asynchron ausgegeben und gibt eine Untermenge von MQMD-Feldern zurück.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DEF_PUT_RESPONSE_TYPE im MQINQ-Aufruf bestimmt.

DistLists (MQLONG)

Gibt an, ob Verteilerlistennachrichten in die Warteschlange gestellt werden können.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Ein Nachrichtenkanalagent (MCA) setzt das Attribut, um den lokalen Warteschlangenmanager zu informieren, ob der Warteschlangenmanager am anderen Ende des Kanals Verteilerlisten unterstützt. Dieser Warteschlangenmanager (*Partner*-Warteschlangenmanager genannt) empfängt die Nachricht, nachdem sie vom sendenden MCA aus der lokalen Übertragungswarteschlange entfernt wurde.

Der sendende MCA setzt das Attribut jedes Mal, wenn er eine Verbindung zum empfangenden MCA auf dem Partnerwarteschlangenmanager herstellt. Dadurch kann der sendende MCA bewirken, dass der lokale Warteschlangenmanager in die Übertragungswarteschlange nur Nachrichten einfügt, die der Partnerwarteschlangenmanager ordnungsgemäß verarbeiten kann.

Dieses Attribut ist hauptsächlich für die Verwendung mit Übertragungswarteschlangen bestimmt, aber die beschriebene Verarbeitung wird ungeachtet der für die Warteschlange definierten Nutzung durchgeführt (siehe Usage-Attribut).

Folgende Werte sind möglich:

MQDL_SUPPORTED
Verteilerlistennachrichten können in der Warteschlange gespeichert und an den Partnerwarteschlangenmanager in dieser Form übertragen werden. Somit wird der erforderliche Verarbeitungsaufwand für das Senden von Nachrichten an mehrere Empfänger reduziert.

MQDL_NOT_SUPPORTED
Verteilerlistennachrichten können nicht in der Warteschlange gespeichert werden, weil der Partnerwarteschlangenmanager keine Verteilerlisten unterstützt. Wenn eine Anwendung eine Verteilerlistennachricht einreiht und diese Nachricht in dieser Warteschlange zu platzieren ist, teilt der Warteschlangenmanager die Verteilerlistennachricht auf und platziert stattdessen die einzelnen Nachrichten in der Warteschlange. Dies erhöht den Verarbeitungsaufwand für das Senden der Nachricht an mehrere Ziele, stellt aber sicher, dass die Nachrichten vom Partner-Warteschlangenmanager ordnungsgemäß verarbeitet werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_DIST_LISTS im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

Dieses Attribut wird unter z/OS nicht unterstützt.

HardenGetBackout (MQLONG)

Für jede Nachricht wird gezählt, wie oft die Nachricht von einem MQGET-Aufruf innerhalb einer Arbeitseinheit abgerufen wird, und diese Arbeitseinheit anschließend zurückgesetzt wird.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Dieser Zähler ist im Feld *BackoutCount* im Nachrichtendeskriptor verfügbar, nachdem der MQGET-Aufruf abgeschlossen wurde.

Der Nachrichtenrücksetzungszähler bleibt bei einem Warteschlangenmanagerneustart erhalten. Um aber sicherzustellen, dass der Zähler korrekt funktioniert, müssen die Informationen bei jedem Abruf einer Nachricht durch den MQGET-Aufruf innerhalb einer Arbeitseinheit für diese Warteschlange *permanent gespeichert* werden (Aufzeichnung auf einer Festplatte oder einem anderen permanenten Speicher). Wenn dieser Vorgang nicht ausgeführt wird, der Warteschlangenmanager fehlschlägt und der MQGET-Aufruf zurückgesetzt wird, wird der Zähler möglicherweise nicht erhöht.

Das permanente Speichern von Informationen bei jedem MQGET-Aufruf in einer Arbeitseinheit bewirkt jedoch zusätzliche Verarbeitungskosten, also sollten Sie das Attribut *HardenGetBackout* nur auf MQQA_BACKOUT_HARDENED setzen, wenn es entscheidend ist, dass der Zähler präzise ist.

Unter IBM i, UNIX und Windows wird der Nachrichtenrücksetzungszähler immer permanent gespeichert, unabhängig von der Einstellung dieses Attributs.

Folgende Werte sind möglich:

MQQA_BACKOUT_HARDENED
Die Aufzeichnung wird verwendet, um sicherzustellen, dass der Rücksetzungszähler für Nachrichten in dieser Warteschlange richtig ist.

MQQA_BACKOUT_NOT_HARDENED
Die Aufzeichnung wird nicht verwendet, um sicherzustellen, dass der Rücksetzungszähler für Nachrichten in dieser Warteschlange richtig ist. Der Wert des Zählers ist daher möglicherweise niedriger, als die korrekte Anzahl.

Der Wert dieses Attributs wird mit dem Selektor MQIA_HARDEN_GET_BACKOUT im MQINQ-Aufruf bestimmt.

IndexType (MQLONG)

Gibt den Typ des Index an, den der Warteschlangenmanager für Nachrichten in der Warteschlange verwaltet.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Der erforderliche Indextyp hängt davon ab, wie die Anwendung Nachrichten abruft und ob die Warteschlange eine gemeinsam genutzte Warteschlange oder eine nicht gemeinsam genutzt Warteschlange ist (siehe QSGDisp-Attribut). Die folgenden Werte sind für *IndexType* möglich:

MQIT_NONE
Für diese Warteschlange verwaltet der Warteschlangenmanager keinen Index. Verwenden Sie diesen Wert für Warteschlangen, die normalerweise sequenziell verarbeitet werden, d. h., ohne Auswahlkriterien im MQGET-Aufruf zu verwenden.

MQIT_MSG_ID
Der Warteschlangenmanager verwaltet einen Index, der die Nachrichten-IDs der Nachrichten in der Warteschlange verwendet. Verwenden Sie diesen Wert für Warteschlangen, bei denen die Anwendung normalerweise Nachrichten mit der Nachrichten-ID als Auswahlkriterium im MQGET-Aufruf verwendet.

MQIT_CORREL_ID
Der Warteschlangenmanager verwaltet einen Index, der die Korrelations-IDs der Nachrichten in der Warteschlange verwendet. Verwenden Sie diesen Wert für Warteschlangen, bei denen die Anwendung normalerweise Nachrichten mit der Korrelations-IDs als Auswahlkriterium im MQGET-Aufruf verwendet.

MQIT_MSG_TOKEN
Der Warteschlangenmanager verwaltet einen Index, der die Nachrichtentoken der Nachrichten in der Warteschlange zur Nutzung der Workload-Manager-Funktionen von z/OS verwendet.

Sie *müssen* diese Option für vom WLM verwaltete Warteschlangen angeben, für andere Warteschlangentypen darf sie nicht angegeben werden. Verwenden Sie diesen Wert auch nicht für eine Warteschlange, bei der die Anwendung die z/OS-Workload-Manager-Funktionen nicht verwendet, jedoch Nachrichten mit dem Nachrichtentoken als Auswahlkriterium im MQGET-Aufruf verwendet.

MQIT_GROUP_ID
Der Warteschlangenmanager verwaltet einen Index, der die Gruppen-IDs der Nachrichten in der Warteschlange verwendet. Dieser Wert *muss* für Warteschlangen verwendet werden, bei denen die Anwendung Nachrichten mit der Option MQGMO_LOGICAL_ORDER im MQGET-Aufruf abruft.

Eine Warteschlange mit diesem Indextyp kann keine Übertragungswarteschlange sein. Eine gemeinsam genutzte Warteschlange mit diesem Indextyp muss so definiert sein, dass sie ein CFSTRUCT-Objekt auf CFLEVEL(3) oder CFLEVEL(4) zuordnet.

Anmerkung:

1. Die physische Reihenfolge von Nachrichten in einer Warteschlange mit Indextyp MQIT_GROUP_ID ist nicht festgelegt, weil die Warteschlange für das effiziente Abrufen von Nachrichten mit der Option MQGMO_LOGICAL_ORDER im MQGET-Aufruf optimiert wird. Das bedeutet, dass die physische Reihenfolge der Nachrichten normalerweise nicht die Reihenfolge ist, in der die Nachrichten in der Warteschlange eintreffen.
2. Bei einer MQIT_GROUP_ID-Warteschlange mit *MsgDeliverySequence* mit dem Wert MQMDS_PRIORITY, verwendet der Warteschlangenmanager die Nachrichtenprioritäten 0 und 1, um das Abrufen von Nachrichten in logischer Reihenfolge zu optimieren. Daher darf die erste Nachricht in einer Gruppe nicht die Priorität null oder eins haben. Sollte dies der Fall sein, wird die Nachricht verarbeitet, als sei die Priorität zwei. Das Feld *Priority* in der MQMD-Struktur wird nicht geändert.

Weitere Informationen zu Nachrichtengruppen finden Sie in der Beschreibung der Gruppen- und Segmentoptionen im Abschnitt MQGMO - Optionsfeld.

Der jeweils zu verwendende Indextyp ist in Tabelle 574 auf Seite 850 und Tabelle 575 auf Seite 851 aufgeführt.

Tabelle 574. Empfohlene oder erforderliche Werte für den WS-Indextyp, wenn MQGMO_LOGICAL_ORDER nicht angegeben ist

Auswahlkriterien beim Aufruf MQGET	**Indextyp für nicht gemeinsam genutzte Warteschlange**	**Indextyp für gemeinsam genutzte Warteschlange**
--	Alle	Alle
Auswahl mit einer ID:		
Nachrichten-ID	MQIT_MSG_ID empfohlen	MQIT_NONE oder MQIT_MSG_ID erforderlich, MQIT_MSG_ID empfohlen
Korrelations-ID	MQIT_CORREL_ID empfohlen	MQIT_CORREL_ID erforderlich
Gruppen-ID	MQIT_GROUP_ID empfohlen	MQIT_GROUP_ID erforderlich
Auswahl mit zwei IDs:		
Nachrichten-ID plus Korrelations-ID	MQIT_MSG_ID oder MQIT_CORREL_ID empfohlen	MQIT_NONE oder MQIT_MSG_ID oder MQIT_CORREL_ID erforderlich (Um die Effizienz zu erhöhen wird empfohlen, den Indextyp auszuwählen, der dem MQMD-Feld mit den meisten eindeutigen Schlüsseln entspricht)
Nachrichten-ID plus Gruppen-ID	MQIT_MSG_ID oder MQIT_GROUP_ID empfohlen	Nicht unterstützt
Korrelations-ID plus Gruppen-ID	MQIT_CORREL_ID oder MQIT_GROUP_ID empfohlen	Nicht unterstützt
Auswahl mit drei IDs:		
Nachrichten-ID plus Korrelations-ID plus Gruppen-ID	MQIT_MSG_ID oder MQIT_CORREL_ID oder MQIT_GROUP_ID empfohlen	Nicht unterstützt
Auswahl mit gruppenbezogenen Kriterien:		
Gruppen-ID plus Nachrichtenfolgenummer	MQIT_GROUP_ID erforderlich	MQIT_GROUP_ID erforderlich
Nachrichtenfolgenummer (muss 1 sein)	MQIT_GROUP_ID erforderlich	MQIT_GROUP_ID erforderlich
Auswahl mit Nachrichtentoken:		
Nachrichtentoken für Anwendungsverwendung	Nicht MQIT_MSG_TOKEN verwenden	
Nachrichtentoken für WLM-Verwendung	MQIT_MSG_TOKEN erforderlich	Nicht unterstützt

Tabelle 575. Empfohlene oder erforderliche Werte für den Warteschlangenindextyp, wenn MQGMO_LOGICAL_ORDER angegeben ist

Auswahlkriterien beim Aufruf MQGET	Indextyp für nicht gemeinsam genutzte Warteschlange	Indextyp für gemeinsam genutzte Warteschlange
--	MQIT_GROUP_ID erforderlich	MQIT_GROUP_ID erforderlich
Auswahl mit einer ID:		
Nachrichten-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt
Korrelations-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt
Gruppen-ID	MQIT_GROUP_ID erforderlich	MQIT_GROUP_ID erforderlich
Auswahl mit zwei IDs:		
Nachrichten-ID plus Korrelations-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt
Nachrichten-ID plus Gruppen-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt
Korrelations-ID plus Gruppen-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt
Auswahl mit drei IDs:		
Nachrichten-ID plus Korrelations-ID plus Gruppen-ID	MQIT_GROUP_ID erforderlich	Nicht unterstützt

Der Wert dieses Attributs wird mit dem Selektor MQIA_INDEX_TYPE im MQINQ-Aufruf ermittelt.

Dieses Attribut wird nur unter z/OS unterstützt.

InhibitGet (MQLONG)

Dadurch wird gesteuert, ob Operationen für diese Warteschlange zulässig sind.

Lokal	Modell	Alias	Fern	Cluster
X	X	X		

Wenn es sich bei der Warteschlange um eine Aliaswarteschlange handelt, müssen die Get-Operationen sowohl für die Alias- als auch die Basiswarteschlange zum Zeitpunkt der Get-Operation zulässig sein, damit der MQGET-Aufruf erfolgreich ausgeführt werden kann. Folgende Werte sind möglich:

MQQA_GET_INHIBITED
Get-Operationen werden unterdrückt.

MQGET-Aufrufe schlagen mit Ursachencode MQRC_GET_INHIBITED fehl. Dies schließt MQGET-Aufrufe ein, bei denen MQGMO_BROWSE_FIRST oder MQGMO_BROWSE_NEXT angegeben ist.

Anmerkung: Wenn ein MQGET-Aufruf in einer Arbeitseinheit erfolgreich abgeschlossen wird, verhindert das nachträgliche Ändern des Werts des Attributs *InhibitGet* auf MQQA_GET_INHIBITED nicht, dass die Arbeitseinheit festgeschrieben wird.

MQQA_GET_ALLOWED
GET-Operationen sind zulässig.

Der Wert dieses Attributs wird mit dem Selektor MQIA_INHIBIT_GET im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

InhibitPut (MQLONG)

Dadurch wird gesteuert, ob Operationen für diese Warteschlange zulässig sind.

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	X

Wenn es im Auflösungspfad des Warteschlangennamens mehrere Definitionen gibt, müssen Put-Operationen bei *jeder* Definition im Pfad (einschließlich aller Warteschlangenmanager-Aliasdefinitionen) zum Zeitpunkt der Operation zulässig sein, damit der MQPUT- oder MQPUT1-Aufruf erfolgreich ist. Folgende Werte sind möglich:

MQQA_PUT_INHIBITED
Put-Operationen werden unterdrückt.

MQPUT- und MQPUT1-Aufrufe schlagen mit Ursachencode MQRC_PUT_INHIBITED fehl.

Anmerkung: Wenn ein MQPUT-Aufruf in einer Arbeitseinheit erfolgreich abgeschlossen wird, verhindert das nachträgliche Ändern des Werts des Attributs *InhibitPut* auf MQQA_PUT_INHIBITED nicht, dass die Arbeitseinheit festgeschrieben wird.

MQQA_PUT_ALLOWED
PUT-Operationen werden zugelassen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_INHIBIT_PUT im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

InitiationQName (MQCHAR48)
Dies ist der Name einer Warteschlange, die auf dem lokalen Warteschlangenmanager definiert ist. Die Warteschlange muss den Typ MQQT_LOCAL haben.

Lokal	Modell	Alias	Fern	Cluster
X				

Der Warteschlangenmanager sendet eine Auslösenachricht an die Initialisierungswarteschlange, wenn ein Anwendungsstart erforderlich ist, weil eine Nachricht in der Warteschlange eintrifft, zu der dieses Attribut gehört. Die Initialisierungswarteschlange muss von einer Auslösemonitoranwendung überwacht werden, die die entsprechende Anwendung nach dem Empfang der Auslösenachricht startet.

Der Wert dieses Attributs wird mit dem Selektor MQCA_INITIATION_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

MaxMsgLength (MQLONG)
Dies ist ein oberer Grenzwert für die Länge der längsten *Physisch*-Nachricht, die in die Warteschlange gestellt werden kann.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Da das Warteschlangenattribut *MaxMsgLength* unabhängig vom Warteschlangenmanagerattribut *MaxMsgLength* festgelegt werden kann, ist die tatsächliche Obergrenze für die längste physische Nachricht, die in die Warteschlange eingefügt werden kann, der niedrigere der beiden Werte.

Wenn der Warteschlangenmanager Segmentierung unterstützt, kann eine Anwendung eine *logische* Nachricht einreihen, die länger als der niedrigere der beiden Werte von *MaxMsgLength* ist, aber nur, wenn die Anwendung das Flag MQMF_SEGMENTATION_ALLOWED in MQMD angibt. Wenn dieses Flag angegeben ist, liegt die Obergrenze für die Länge einer logischen Nachricht bei 999 999 999 Byte. In der Regel führen jedoch vom Betriebssystem oder der Umgebung, in der die Anwendung ausgeführt wird, vorgegebene Ressourcenbeschränkungen zu einem niedrigeren Grenzwert.

Der Versuch, in die Warteschlange eine Nachricht einzufügen, die zu lang ist, schlägt mit einem der folgenden Ursachencodes fehl:

- MQRC_MSG_TOO_BIG_FOR_Q, wenn die Nachricht für die Warteschlange zu groß ist

- MQRC_MSG_TOO_BIG_FOR_Q_MGR, wenn die Nachricht für den Warteschlangenmanager, aber nicht für die Warteschlange zu groß ist

Die Untergrenze für das Attribut *MaxMsgLength* ist null die Obergrenze ist 100 MB (104 857 600 Byte).

Weitere Informationen finden Sie im Abschnitt MQPUT - BufferLength-Parameter.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_MSG_LENGTH im MQINQ-Aufruf bestimmt.

MaxQDepth (MQLONG)

Dies ist der definierte obere Grenzwert für die Anzahl der physischen Nachrichten, die in der Warteschlange zu einem beliebigen Zeitpunkt vorhanden sein können.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Der Versuch, eine Nachricht in eine Warteschlange einzureihen, die bereits *MaxQDepth* Nachrichten enthält, schlägt mit Ursachencode MQRC_Q_FULL fehl.

Sowohl die Verarbeitung der Arbeitseinheit als auch die Segmentierung der Nachrichten können dazu führen, dass die tatsächliche Anzahl der physischen Nachrichten in der Warteschlange den Wert für *MaxQDepth* überschreitet. Dies wirkt sich allerdings nicht auf die Abrufbarkeit der Nachrichten aus; *alle* Nachrichten in der Warteschlange können mit dem MQGET-Aufruf abgerufen werden.

Der Wert dieses Attributs ist null oder größer. Die Obergrenze wird von der Umgebung bestimmt:

- Unter AIX, HP-UX, z/OS, Solaris, Linuxund Windowsdarf der Wert 999 999 999 nicht überschreiten.
- Unter IBM i darf der Wert 640 000 nicht überschreiten.

Anmerkung: Der für die Warteschlange verfügbare Speicherplatz kann ausgeschöpft sein, auch wenn es weniger als *MaxQDepth* Nachrichten in der Warteschlange gibt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MAX_Q_DEPTH im MQINQ-Aufruf bestimmt.

MsgDeliverySequence (MQLONG)

Lokal	Modell	Alias	Fern	Cluster
X	X			

Dieses Attribut bestimmt die Reihenfolge, in der der MQGET-Aufruf der Anwendung Nachrichten zurückgibt:

MQMDS_FIFO
Nachrichten werden in der Reihenfolge First In/First Out (FIFO) zurückgegeben.

Ein MQGET-Aufruf gibt die *erste* Nachricht zurück, die die im Aufruf angegebenen Auswahlkriterien erfüllt, ungeachtet der Priorität der Nachricht.

MQMDS_PRIORITY
Nachrichten werden in der Reihenfolge ihrer Priorität zurückgegeben.

Ein MQGET-Aufruf gibt die Nachricht *mit höchster Priorität* zurück, die die im Aufruf angegebenen Auswahlkriterien erfüllt. Innerhalb der einzelnen Prioritätsebenen werden die Nachrichten in der Reihenfolge First In/First Out (FIFO) zurückgegeben.

- Unter z/OS: Bei Warteschlangen mit dem *IndexType* MQIT_GROUP_ID gibt das Attribut *MsgDeliverySequence* die Reihenfolge an, in der Nachrichtengruppen an die Anwendung zurückgegeben werden. Die jeweilige Reihenfolge, in der die Gruppen zurückgegeben werden, wird von der Position oder Priorität der ersten Nachricht in jeder Gruppe bestimmt. Die physische Reihenfolge von Nachrichten in der Warteschlange ist nicht festgelegt, weil die Warteschlange für effizientes Abrufen von Nachrichten mit der Option MQGMO_LOGICAL_ORDER im MQGET-Aufruf optimiert wird.
- Unter z/OS: Beim *IndexType* MQIT_GROUP_ID und *MsgDeliverySequence* MQMDS_PRIORITY verwendet der Warteschlangenmanager die Nachrichtenprioritäten null und eins, um das Abrufen von

Nachrichten in logischer Reihenfolge zu optimieren. Daher darf die erste Nachricht in einer Gruppe nicht die Priorität null oder eins haben. Sollte dies der Fall sein, wird die Nachricht verarbeitet, als sei die Priorität zwei. Das Feld *Priority* in der MQMD-Struktur wird nicht geändert.

Wenn die entsprechenden Attribute geändert werden, während sich Nachrichten in der Warteschlange befinden, ist die Reihenfolge der Übermittlung wie folgt:

- Die Reihenfolge, in der Nachrichten vom MQGET-Aufruf zurückgegeben werden, wird von den Werten der Attribute *MsgDeliverySequence* und *DefPriority* bestimmt, die zum Zeitpunkt gelten, zu dem die Nachricht in der Warteschlange eintrifft:
  - Wenn *MsgDeliverySequence* MQMDS_FIFO ist, wenn die Nachricht eintrifft, wird die Nachricht in die Warteschlange eingefügt, als sei ihre Priorität *DefPriority*. Dies wirkt sich nicht auf den Wert des Felds *Priority* im Nachrichtendeskriptor der Nachricht aus. Dieses Feld behält den Wert bei, den es hatte, als die Nachricht erstmals eingereiht wurde.
  - Wenn *MsgDeliverySequence* MQMDS_PRIORITY ist, wenn die Nachricht eintrifft, wird die Nachricht in die Warteschlange an der Stelle eingefügt, die der Priorität entspricht, die durch das Feld *Priority* im Nachrichtendeskriptor angegeben ist.

  Wenn der Wert des Attributs *MsgDeliverySequence* geändert wird, während sich Nachrichten in der Warteschlange befinden, hat dies keine Auswirkung auf die Reihenfolge der bereits eingereihten Nachrichten.

  Wenn der Wert des Attributs *DefPriority* geändert wird, während sich Nachrichten in der Warteschlange befinden, werden die Nachrichten nicht notwendigerweise in der FIFO-Reihenfolge übergeben, auch wenn das Attribut *MsgDeliverySequence* auf MQMDS_FIFO festgelegt ist. Nachrichten, die mit höherer Priorität in die Warteschlange eingefügt wurden, werden zuerst bereitgestellt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MSG_DELIVERY_SEQUENCE im MQINQ-Aufruf bestimmt.

NonPersistentMessageClass (MQLONG)

Das Zuverlässigkeitsziel für nicht persistente Nachrichten.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Dieses Attribut gibt an, wann in diese Warteschlange eingereihte nicht persistente Nachrichten gelöscht werden.

MQNPM_CLASS_NORMAL
Nicht persistente Nachrichten sind auf die Laufzeit der Warteschlangenmanager-Sitzung beschränkt, bei einem Warteschlangenmanagerneustart werden die Nachrichten verworfen. Dieser Wert ist nur gültig für nicht gemeinsam genutzte Warteschlangen, es ist der Standardwert.

MQNPM_CLASS_HIGH
Der Warteschlangenmanager versucht, nicht persistente Nachrichten für die Laufzeit der Warteschlange beizubehalten. Nicht persistente Nachrichten können dennoch bei einem Ausfall verloren gehen. Dieser Wert wird für gemeinsam genutzte Warteschlangen erzwungen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_NPM_CLASS im MQINQ-Aufruf bestimmt.

OpenInputCount (MQLONG)

Dies ist die Anzahl der Kennungen, die derzeit für das Entfernen von Nachrichten aus der Warteschlange mit Hilfe des MQGET-Aufrufs gültig sind.

Lokal	Modell	Alias	Fern	Cluster
X				

Es handelt sich um die Gesamtzahl dieser Kennungen, die dem *lokalen* Warteschlangenmanager bekannt ist. Wenn es sich bei der Warteschlange um eine gemeinsam genutzte Warteschlange handelt, bezieht die Anzahl nicht die Öffnungen zur Eingabe ein, die für die Warteschlange bei anderen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört, ausgeführt werden.

Der Zähler umfasst Kennungen, bei denen eine Aliaswarteschlange, die in diese Warteschlange aufgelöst wird, zur Eingabe geöffnet wurde. Der Zähler umfasst keine Kennungen, bei denen die Warteschlange für Aktionen ohne Eingabe geöffnet wurde (z. B. eine Warteschlange wurde nur zum Durchsuchen geöffnet).

Der Wert dieses Attributs ändert sich, während der Warteschlangenmanager ausgeführt wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_OPEN_INPUT_COUNT im MQINQ-Aufruf bestimmt.

OpenOutputCount (MQLONG)

Dies ist die Anzahl der Kennungen, die derzeit für das Hinzufügen von Nachrichten zur Warteschlange mit Hilfe des MQPUT-Aufrufs gültig sind.

Lokal	Modell	Alias	Fern	Cluster
X				

Es handelt sich um die Gesamtzahl dieser Kennungen, die dem *lokalen* Warteschlangenmanager bekannt ist; diese bezieht nicht die Öffnungen zur Ausgabe ein, die im fernen Warteschlangenmanager für diese Warteschlange ausgeführt werden. Wenn es sich bei der Warteschlange um eine gemeinsam genutzte Warteschlange handelt, bezieht die Anzahl nicht die Öffnungen zur Ausgabe ein, die für die Warteschlange bei anderen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange, zu der der lokale Warteschlangenmanager gehört, ausgeführt werden.

Der Zähler umfasst Kennungen, bei denen eine Aliaswarteschlange, die in diese Warteschlange aufgelöst wird, zur Ausgabe geöffnet wurde. Der Zähler umfasst keine Kennungen, bei denen die Warteschlange für Aktionen ohne Ausgabe geöffnet wurde (z. B. eine Warteschlange wurde nur zum Abfragen geöffnet).

Der Wert dieses Attributs ändert sich, während der Warteschlangenmanager ausgeführt wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_OPEN_OUTPUT_COUNT im MQINQ-Aufruf bestimmt.

ProcessName (MQCHAR48)

Dies ist der Name eines Prozessobjekts, das im lokalen Warteschlangenmanager definiert ist. Das Prozessobjekt gibt ein Programm an, das die Warteschlange verarbeiten kann.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Der Wert dieses Attributs wird mit dem Selektor MQCA_PROCESS_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_PROCESS_NAME_LENGTH angegeben.

PropertyControl (MQLONG)

Gibt an, wie Nachrichteneigenschaften für Nachrichten verarbeitet werden, die mit dem MQGET-Aufruf und der Option MQGMO_PROPERTIES_AS_Q_DEF aus Warteschlangen abgerufen werden.

Lokal	Modell	Alias	Fern	Cluster
X	X	X		

Folgende Werte sind möglich:

MQPROP_ALL
Alle Eigenschaften der Nachricht werden in die Nachricht einbezogen, wenn diese an die Anwendung gesendet wird. Die Eigenschaften werden in einem oder mehreren MQRFH2-Headern in den Nachrichtendaten eingeschlossen, ausgenommen der Eigenschaften im Nachrichtendeskriptor (oder Erweiterung). Wenn eine Nachrichtenkennung angegeben ist, dann werden die Eigenschaften in der Nachrichtenkennung zurückgegeben.

MQPROP_COMPATIBILITY
Wenn die Nachricht eine Eigenschaft mit einem der Präfixe mcd., Jms. usr. oder mqext. enthält, werden alle Nachrichteneigenschaften der Anwendung in einem MQRFH2-Header zugestellt: Andernfalls werden alle Eigenschaften der Nachricht, außer denen, die im Nachrichtendeskriptor (oder Erweiterung) enthalten sind, gelöscht und sind nicht mehr für die Anwendung verfügbar. Das ist der Standardwert. Er ermöglicht Anwendungen, die mit JMS zusammengehörige Eigenschaften in einem MQRFH2-Header in den Nachrichtendaten erwarten, unverändert fortzufahren. Wenn eine Nachrichtenkennung angegeben ist, dann werden die Eigenschaften in der Nachrichtenkennung zurückgegeben.

MQPROP_FORCE_MQRFH2
Unabhängig davon, ob die Anwendung eine Nachrichtenkennung angibt, werden die Eigenschaften immer in den Nachrichtendaten in einem MQRFH2-Header zurückgegeben. Eine gültige Nachrichtenkennung, die im Feld "MsgHandle" der MQGMO-Struktur im MQGET-Aufruf angegeben wird, wird ignoriert. Die Eigenschaften der Nachricht sind nicht über die Nachrichtenkennung zugänglich.

MQPROP_NONE
Alle Eigenschaften der Nachricht, außer denen im Nachrichtendeskriptor (oder Erweiterung), werden von der Nachricht entfernt, bevor die Nachricht an die Anwendung gesendet wird. Wenn eine Nachrichtenkennung angegeben ist, dann werden die Eigenschaften in der Nachrichtenkennung zurückgegeben.

Dieser Parameter ist gültig für lokale Warteschlangen, Alias- und Modellwarteschlangen. Der zugehörige Wert wird mit dem Selektor MQIA_PROPERTY_CONTROL im MQINQ-Aufruf bestimmt.

QDepthHighEvent (MQLONG)
Dadurch wird gesteuert, ob Ereignisse vom Typs "Queue Depth High" generiert werden.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Ein 'Warteschlangenlänge hoch'-Ereignis zeigt an, dass eine Anwendung eine Nachricht in eine Warteschlange eingereiht hat, wodurch die Anzahl der Nachrichten in der Warteschlange den oberen Schwellenwert für die Warteschlangenlänge erreicht oder überschritten hat (siehe Attribut *QDepthHighLimit*).

Anmerkung: Der Wert dieses Attributs kann sich dynamisch ändern.

Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_DEPTH_HIGH_EVENT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QDepthHighLimit (MQLONG)
Dies ist der Schwellenwert, mit dem die Warteschlangenlänge verglichen wird, um ein Ereignis mit hoher Warteschlangenlänge zu generieren.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Dieses Ereignis signalisiert, dass eine Anwendung eine Nachricht in eine Warteschlange gestellt hat und damit die Nachrichtenanzahl in der Warteschlange größer oder gleich der Obergrenze für die Warteschlangenlänge ist. Siehe Attribut QDepthHighEvent.

Der Wert wird als Prozentsatz der maximalen Warteschlangenlänge (Attribut *MaxQDepth*) angegeben und ist größer-gleich null oder kleiner-gleich 100. Der Standardwert ist 80.

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_DEPTH_HIGH_LIMIT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QDepthLowEvent (MQLONG)

Dadurch wird gesteuert, ob Ereignisse mit Warteschlangenlänge (Queue Depth Low) generiert werden

Lokal	Modell	Alias	Fern	Cluster
X	X			

Ein 'Warteschlangenlänge niedrig'-Ereignis zeigt an, dass eine Anwendung eine Nachricht aus einer Warteschlange abgerufen hat, wodurch die Anzahl Nachrichten in der Warteschlange den unteren Schwellenwert für die Warteschlangenlänge erreicht oder unterschritten hat (siehe Attribut QDepthLowLimit).

Anmerkung: Der Wert dieses Attributs kann sich dynamisch ändern.

Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_DEPTH_LOW_EVENT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QDepthLowLimit (MQLONG)

Dies ist der Schwellenwert, mit dem die Warteschlangentiefe verglichen wird, um ein Ereignis „Warteschlangentiefe niedrig“ zu erzeugen.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Dieses Ereignis signalisiert, dass eine Nachricht von einer Anwendung aus einer Warteschlange abgerufen wurde und damit die Anzahl Nachrichten in der Warteschlange kleiner-gleich der Untergrenze für die Warteschlangenlänge ist. Siehe Attribut QDepthLowEvent.

Der Wert wird als Prozentsatz der maximalen Warteschlangenlänge (Attribut *MaxQDepth*) angegeben und ist größer-gleich null oder kleiner-gleich 100. Der Standardwert ist 20.

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_DEPTH_LOW_LIMIT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QDepthMaxEvent (MQLONG)

Dieses Attribut gibt an, ob 'Warteschlangenlänge voll'-Ereignisse generiert werden. Ein Ereignis „Queue Full“ gibt an, dass ein Einreihungsvorgang für eine Warteschlange zurückgewiesen wurde, weil die Warteschlange voll ist, d. h. die Warteschlangenlänge hat ihren Maximalwert bereits erreicht.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Anmerkung: Der Wert dieses Attributs kann sich dynamisch ändern.

Folgende Werte sind möglich:

MQEVR_DISABLED
Ereignisberichterstellung inaktiviert.

MQEVR_ENABLED
Ereignisberichterstellung aktiviert

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_DEPTH_MAX_EVENT im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QDesc (MQCHAR64)

Verwenden Sie dieses Feld für einen beschreibenden Kommentar.

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	X

Der Feldinhalt ist für den Warteschlangenmanager nicht von Bedeutung, es ist aber erforderlich, dass das Feld nur solche Zeichen enthält, die angezeigt werden können. Das Feld darf keine Nullzeichen enthalten; ggf. wird es rechts mit Leerzeichen aufgefüllt. In einer DBCS-Installation kann das Feld DBCS-Zeichen enthalten (die maximale Länge beträgt 64 Byte).

Anmerkung: Wenn das Feld Zeichen enthält, die nicht Bestandteil des Zeichensatzes des Warteschlangenmanagers sind (gemäß der Definition im Warteschlangenmanagerattribut *CodedCharSetId*), werden sie möglicherweise falsch übersetzt, wenn das Feld an einen anderen Warteschlangenmanager gesendet wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_Q_DESC im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_DESC_LENGTH angegeben.

QName (MQCHAR48)

Dies ist der Name einer Warteschlange, die auf dem lokalen Warteschlangenmanager definiert ist.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Alle Warteschlangen, die in einem Warteschlangenmanager definiert sind, nutzen gemeinsam denselben Warteschlangennamensbereich. Deshalb können eine MQQT_LOCAL-Warteschlange und eine MQQT_ALIAS-Warteschlange nicht denselben Namen haben.

Der Wert dieses Attributs wird mit dem Selektor MQCA_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

QServiceInterval (MQLONG)
Dies ist das Serviceintervall, das zum Vergleich verwendet wird, um die Ereignisse Serviceintervall hoch und Serviceintervall OK zu erzeugen.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Siehe Attribut QServiceIntervalEvent.

Der Wert wird in Millisekunden angegeben und ist größer-gleich null oder kleiner-gleich 999 999 999.

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_SERVICE_INTERVAL im MQINQ-Aufruf bestimmt.

Dieses Attribut wird unter z/OS unterstützt, aber der Wert kann nicht mit dem MQINQ-Aufruf bestimmt werden.

QServiceIntervalEvent (MQLONG)
Dadurch wird gesteuert, ob Ereignisse des Typs „Service Interval High“ oder „Service Interval OK“ generiert werden

Lokal	Modell	Alias	Fern	Cluster
X	X			

- Das Ereignis "Service Interval High" wird generiert, wenn eine Prüfung ergibt, dass mindestens für den Zeitraum, der im Attribut *QServiceInterval* angegebenen wird, keine Nachrichten aus der Warteschlange abgerufen wurden.
- Das Ereignis "Service Interval OK" wird generiert, wenn eine Prüfung ergibt, dass innerhalb des Zeitraums, der im Attribut *QServiceInterval* angegebenen wird, Nachrichten aus der Warteschlange abgerufen wurden.

Anmerkung: Der Wert dieses Attributs kann sich dynamisch ändern.

Folgende Werte sind möglich:

MQQSIE_HIGH
Ereignisse "Queue Service Interval High" sind aktiviert.

- 'Warteschlangenserviceintervall hoch'-Ereignisse **aktiviert** und
- 'Warteschlangenserviceintervall OK'-Ereignisse **inaktiviert**.

MQQSIE_OK
Ereignisse "Queue Service Interval OK" sind aktiviert.

- 'Warteschlangenserviceintervall hoch'-Ereignisse **inaktiviert** und
- 'Warteschlangenserviceintervall OK'-Ereignisse **aktiviert**.

MQQSIE_NONE
Keine der Ereignisse "Queue Service Interval" sind aktiviert.

- 'Warteschlangenserviceintervall hoch'-Ereignisse **inaktiviert** und
- 'Warteschlangenserviceintervall OK'-Ereignisse ebenfalls **inaktiviert**

Bei gemeinsam genutzten Warteschlangen wird der Wert dieses Attributs ignoriert und der Wert MQQSIE_NONE angenommen.

Weitere Informationen zu Ereignissen finden Sie unter Ereignisüberwachung .

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_SERVICE_INTERVAL_EVENT im MQINQ-Aufruf bestimmt.

Unter z/OS kann der Wert dieses Attributs nicht mit dem MQINQ-Aufruf bestimmt werden.

QSGDisp (MQLONG)

Gibt die Disposition der Warteschlange an.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	

Folgende Werte sind möglich:

MQQSGD_Q_MGR
Das Objekt hat Warteschlangenmanager-Disposition. Dies bedeutet, dass die Objektdefinition nur dem lokalen Warteschlangenmanager bekannt ist. Anderen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange ist die Definition nicht bekannt.

Jeder Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann ein Objekt mit demselben Namen und Typ wie das aktuelle Objekt haben, aber es handelt sich um separate Objekte ohne Korrelation. Auch die Attribute müssen nicht dieselben sein.

MQQSGD_COPY
Das Objekt ist eine lokale Kopie einer Master-Objektdefinition, die im gemeinsam genutzten Repository existiert. Jeder einzelne Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann über eine eigene Kopie des Objekts verfügen. Anfänglich haben alle Kopien die gleichen Attribute, aber mit MQSC-Befehlen können Sie jede Kopie ändern, wodurch sich die jeweiligen Attribute von denen der anderen Kopien unterscheiden. Die Attribute der Kopien werden resynchronisiert, wenn die Master-Definition im gemeinsam genutzten Repository geändert wird.

MQQSGD_SHARED
Das Objekt weist eine gemeinsam genutzte Disposition auf. Das bedeutet, dass im gemeinsam genutzten Repository eine Einzelinstanz des Objekts vorhanden ist, die allen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange bekannt ist. Wenn ein Warteschlangenmanager in der Gruppe auf das Objekt zugreift, so greift er auf die gemeinsam genutzte Einzelinstanz des Objekts zu.

Der Wert dieses Attributs wird mit dem Selektor MQIA_QSG_DISP im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

QueueAccounting (MQLONG)

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	

Dieses Attribut steuert die Erfassung von Abrechnungsdaten für die Warteschlange. Damit Abrechnungsdaten für eine Warteschlange erfasst werden können, muss auch für die Verbindung die Erfassung von Abrechnungsdaten aktiviert werden, entweder mit dem Warteschlangenmanagerattribut ACCTQ oder über die Optionsfelder in der MQCNO-Struktur im MQCONNX-Aufruf.

Das Attribut hat einen der folgenden Werte:

MQMON_Q_MGR
Abrechnungsdaten für diese Warteschlange werden entsprechend der Einstellung des Warteschlangenmanagerattributs ACCTQ erfasst. Dies ist die Standardeinstellung.

MQMON_OFF
Abrechnungsdaten für diese Warteschlange nicht erfassen.

MQMON_ON
Abrechnungsdaten für diese Warteschlange erfassen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_ACCOUNTING_Q im Aufruf MQINQ bestimmt.

QueueMonitoring (MQLONG)

Steuert die Erfassung von Onlineüberwachungsdaten für Warteschlangen.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Folgende Werte sind möglich:

MQMON_Q_MGR
Überwachungsdaten entsprechend der Einstellung des Warteschlangenmanagerattributs *QueueMonitoring* erfassen. Dies ist der Standardwert.

MQMON_OFF
Die Erfassung von Onlineüberwachungsdaten wird für diese Warteschlange inaktiviert

MQMON_LOW
Wenn der Wert des Warteschlangenmanagerattributs *QueueMonitoring* nicht auf MQMON_NONE gesetzt ist, ist die Erfassung von Onlineüberwachungsdaten für diese Warteschlange mit einer geringen Erfassungsrate aktiviert.

MQMON_MEDIUM
Wenn der Wert des Warteschlangenmanagerattributs *QueueMonitoring* nicht auf MQMON_NONE gesetzt ist, ist die Erfassung von Onlineüberwachungsdaten für diese Warteschlange mit einer mittleren Erfassungsrate aktiviert.

MQMON_HIGH
Wenn der Wert des Warteschlangenmanagerattributs *QueueMonitoring* nicht auf MQMON_NONE gesetzt ist, ist die Erfassung von Onlineüberwachungsdaten für diese Warteschlange mit einer hohen Erfassungsrate aktiviert.

Der Wert dieses Attributs wird mit dem Selektor MQIA_MONITORING_Q im MQINQ-Aufruf bestimmt.

QueueStatistics (MQCHAR12)

Lokal	Modell	Alias	Fern	Cluster
X	X	X	X	

Dieses Attribut steuert die Erfassung von Statistikdaten für Warteschlangen.

Das Attribut hat einen der folgenden Werte:

MQMON_Q_MGR
Abrechnungsdaten für diese Warteschlange werden entsprechend der Einstellung des Warteschlangenmanagerattributs STATQ erfasst. Dies ist die Standardeinstellung.

MQMON_OFF
Erfassung statistischer Daten für diese Warteschlange ausschalten

MQMON_ON
Erfassung statistischer Daten für diese Warteschlange einschalten

QType (MQLONG)

Lokal	Modell	Alias	Fern	Cluster
X		X	X	X

Dieses Attribut gibt den Warteschlangentyp an, er hat einen der folgenden Werte:

MQQT_ALIAS
Aliaswarteschlangendefinition

MQQT_CLUSTER
Clusterwarteschlange.

MQQT_LOCAL
Lokale Warteschlange.

MQQT_REMOTE
Lokale Definition einer fernen Warteschlange.

Der Wert dieses Attributs wird mit dem Selektor MQIA_Q_TYPE im MQINQ-Aufruf bestimmt.

RemoteQMgrName (MQCHAR48)

Lokal	Modell	Alias	Fern	Cluster
			X	

Dies ist der Name des fernen Warteschlangenmanagers, in dem die Warteschlange *RemoteQName* definiert ist. Wenn bei der Warteschlange *RemoteQName* Der Wert des Attributs *QSGDisp* MQQSGD_COPY oder MQQSGD_SHARED ist, kann *RemoteQMgrName* der Name der Gruppe mit gemeinsamer Warteschlange sein, zu der *RemoteQName* gehört.

Wenn eine Anwendung die lokale Definition einer fernen Warteschlange öffnet, darf *RemoteQMgrName* nicht leer sein und nicht den Namen des lokalen Warteschlangenmanagers enthalten. Wenn *XmitQName* nicht belegt ist, wird eine lokale Warteschlange mit demselben Namen, der in *RemoteQMgrName* enthalten ist, als Übertragungswarteschlange verwendet. Wenn es keine Warteschlange mit dem Namen *RemoteQMgrName* gibt, wird die Warteschlange verwendet, die durch das Warteschlangenmanagerattribut *DefXmitQName* angegeben ist.

Wenn diese Definition für einen Warteschlangenmanager-Aliasnamen verwendet wird, ist *RemoteQMgrName* der Name des Warteschlangenmanagers, der den Aliasnamen erhält. Dies kann der Name des lokalen Warteschlangenmanagers sein. Wenn beim Öffnen *XmitQName* hingegen leer ist, muss es eine lokale Warteschlange mit demselben Namen geben, wie in *RemoteQMgrName* angegeben, diese Warteschlange wird als die Übertragungswarteschlange verwendet.

Wird diese Definition für ein Antwortalias verwendet, ist dieser Name der Warteschlangenmanagername, dem *ReplyToQMgr* zugeordnet wird.

Anmerkung: Der für dieses Attribut angegebene Wert wird nicht überprüft, wenn die Warteschlangendefinition erstellt oder geändert wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_REMOTE_Q_MGR_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

RemoteQName (MQCHAR48)

Lokal	Modell	Alias	Fern	Cluster
			X	

Dies ist der Name der Warteschlange, der dem fernen Warteschlangenmanager *RemoteQMgrName* bekannt ist.

Wenn eine Anwendung die lokale Definition einer fernen Warteschlange öffnet, darf *RemoteQName* beim Öffnen nicht leer sein.

Wenn diese Definition für eine Warteschlangenmanager-Aliasdefinition verwendet wird, muss *RemoteQName* beim Öffnen leer sein.

Wird die Definition für ein Antwortalias verwendet, ist dies der Name der Warteschlange, der *ReplyToQ* zugeordnet wird.

Anmerkung: Der für dieses Attribut angegebene Wert wird nicht überprüft, wenn die Warteschlangendefinition erstellt oder geändert wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_REMOTE_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

RetentionInterval (MQLONG)

Dieses Attribut gibt an, für welchen Zeitraum die Warteschlange beibehalten werden soll. Nachdem diese Zeit abgelaufen ist, kann die Warteschlange gelöscht werden.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Die Zeit wird ab dem Zeitpunkt (Datum und Uhrzeit), zu dem die Warteschlange erstellt wurde, in Stunden gemessen. Das Erstellungsdatum und die Erstellungszeit der Warteschlange werden in den Attributen *CreationDate* und *CreationTime* erfasst.

Anhand dieser Informationen kann eine Systemverwaltungsanwendung oder der Bediener Warteschlangen, die nicht mehr erforderlich sind, ermitteln und löschen.

Anmerkung: Er erfolgt keine Aktion des Warteschlangenmanagers, um Warteschlangen basierend auf diesem Attribut zu löschen oder das Löschen von Warteschlangen mit einem noch nicht abgelaufenen Aufbewahrungsintervall zu verhindern, die erforderlichen Aktionen müssen vom Benutzer durchgeführt werden.

Verwenden Sie ein realistisches Aufbewahrungsintervall, um das Ansammeln von persistenten dynamischen Warteschlangen zu verhindern (siehe Attribut DefinitionType). Dieses Attribut kann aber auch mit vordefinierten Warteschlangen verwendet werden.

Der Wert dieses Attributs wird mit dem Selektor MQIA_RETENTION_INTERVAL im MQINQ-Aufruf bestimmt.

Scope (MQLONG)

Dadurch wird gesteuert, ob ein Eintrag für diese Warteschlange auch in einem Zellenverzeichnis vorhanden ist.

Lokal	Modell	Alias	Fern	Cluster
X		X	X	

Ein Zellenverzeichnis wird von einem installierbaren Namensservice bereitgestellt. Folgende Werte sind möglich:

MQSCO_Q_MGR
Die Warteschlangendefinition gilt für den Warteschlangenmanager-Bereich: Sie gilt nur für den Warteschlangenmanager, zu dem sie gehört. Um die Warteschlange eines anderen Warteschlangenmanagers für Ausgaben zu öffnen, muss entweder der Name des betreffenden Warteschlangenmanagers angegeben werden oder der andere Warteschlangenmanager muss über eine lokale Definition der Warteschlange verfügen.

MQSCO_CELL
Die Warteschlangendefinition gilt für den Zellenbereich: Sie wird auch in ein Zellenverzeichnis eingefügt, das allen Warteschlangenmanagern in der Zelle verfügbar ist. Die Warteschlange kann zur Ausgabe von allen Warteschlangenmanagern in der Zelle durch Angabe des Warteschlangennamens geöffnet werden. Die Angabe des Namens des Warteschlangenmanagers, dem die Warteschlange zugeordnet ist, ist nicht erforderlich. Die Warteschlangendefinition ist allerdings keinem Warteschlangenmanager in der Zelle verfügbar, der auch eine lokale Definition einer Warteschlange mit diesem Namen hat, weil die lokale Definition Vorrang hat.

Ein Zellenverzeichnis wird von einem installierbaren Namensservice bereitgestellt.

Das Modell und die dynamischen Warteschlangen können keinen Zellenbereich haben.

Dieser Wert ist nur gültig, wenn ein Namensservice konfiguriert wurde, der ein Zellenverzeichnis unterstützt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SCOPE im MQINQ-Aufruf bestimmt.

Die Unterstützung für dieses Attribut unterliegt folgenden Einschränkungen:

- Unter IBM i wird das Attribut unterstützt, aber nur MQSCO_Q_MGR ist gültig.
- Unter z/OS wird das Attribut nicht unterstützt.

Shareability (MQLONG)
Hier wird angegeben, ob die Warteschlange mehrmals gleichzeitig zur Eingabe geöffnet werden kann.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Folgende Werte sind möglich:

MQQA_SHAREABLE
Warteschlange ist gemeinsam nutzbar.

Mehrfache Öffnung mit der Option MQOO_INPUT_SHARED ist zulässig.

MQQA_NOT_SHAREABLE
Warteschlange ist nicht gemeinsam nutzbar.

Ein MQOPEN-Aufruf mit der Option MQOO_INPUT_SHARED wird als MQOO_INPUT_EXCLUSIVE behandelt.

Der Wert dieses Attributs wird mit dem Selektor MQIA_SHAREABILITY im MQINQ-Aufruf bestimmt.

StorageClass (MQCHAR8)
Dies ist ein benutzerdefinierter Name, mit dem der physische Speicher für die Warteschlange definiert wird. In der Praxis wird eine Nachricht nur dann auf die Festplatte geschrieben, wenn sie aus ihrem Speicherpuffer ausgelagert werden muss.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Der Wert dieses Attributs wird mit dem Selektor MQCA_STORAGE_CLASS im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_STORAGE_CLASS_LENGTH angegeben.

Dieses Attribut wird nur unter z/OS unterstützt.

TriggerControl (MQLONG)
Dadurch wird gesteuert, ob Auslösenachrichten in eine Initialisierungswarteschlange geschrieben werden, um eine Anwendung für den Service der Warteschlange zu starten.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Folgende Werte sind möglich:

MQTC_OFF
Für die Warteschlange werden keine Auslösenachrichten geschrieben. Der Wert von *TriggerType* ist in diesem Fall irrelevant.

MQTC_ON
Es werden Auslösenachrichten für diese Warteschlange geschrieben, wenn die entsprechenden Auslöserereignisse auftreten.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_CONTROL im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

TriggerData (MQCHAR64)
Dies sind Daten mit freiem Format, die der Warteschlangenmanager in die Auslösenachricht einfügt, wenn eine Nachricht, die in dieser Warteschlange eintrifft, dazu führt, dass eine Auslösenachricht in die Initialisierungswarteschlange geschrieben wird.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Der Inhalt dieser Daten hat keine Signifikanz für den Warteschlangenmanager. Sie sind für eine Auslösemonitoranwendung bestimmt, die die Initialisierungswarteschlange verarbeitet, oder für die Anwendung, die der Auslösemonitor startet.

Die Zeichenfolge darf keine Nullen enthalten. Sie wird ggf. rechts mit Leerzeichen aufgefüllt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_TRIGGER_DATA im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf. Die Länge dieses Attributs wird durch MQ_TRIGGER_DATA_LENGTH angegeben.

TriggerDepth (MQLONG)

Lokal	Modell	Alias	Fern	Cluster
X	X			

Gibt die Anzahl an Nachrichten mit der Priorität *TriggerMsgPriority* oder größer an, die in der Warteschlange vorhanden sein muss, bevor eine Auslösenachricht geschrieben wird. Dies gilt, wenn *TriggerType* auf MQTT_DEPTH festgelegt ist. Der Wert von *TriggerDepth* ist größer-gleich eins. Das Attribut wird nicht anderweitig verwendet.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_DEPTH im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

TriggerMsgPriority (MQLONG)
Dies ist die Nachrichtenpriorität, unter der Nachrichten nicht zur Generierung von Auslösenachrichten beitragen (d. h. der Warteschlangenmanager ignoriert diese Nachrichten, wenn er entscheidet, ob eine Auslösenachricht generiert werden soll).

Lokal	Modell	Alias	Fern	Cluster
X	X			

TriggerMsgPriority kann im Bereich von null (Minimum) bis *MaxPriority* (Maximum, siehe Attribut MaxPriority) liegen. Der Wert null bewirkt, dass alle Nachrichten zur Generierung von Auslösenachrichten beitragen.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_MSG_PRIORITY im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

TriggerType (MQLONG)
Dadurch werden die Bedingungen gesteuert, unter denen Auslösenachrichten als Ergebnis von Nachrichten geschrieben werden, die in dieser Warteschlange eintreffen.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Es entspricht einem der folgenden Werte:

MQTT_NONE
: Es werden keine Auslösenachrichten infolge von Nachrichten geschrieben, die in dieser Warteschlange eintreffen. Dies entspricht der Einstellung von *TriggerControl* auf MQTC_OFF.

MQTT_FIRST
Eine Auslösenachricht wird geschrieben, wenn sich die Anzahl der Nachrichten mit der Priorität *TriggerMsgPriority* oder höher in der Warteschlange von 0 auf 1 ändert.

MQTT_EVERY
Eine Auslösenachricht wird geschrieben, wenn in der Warteschlange eine Nachricht der Priorität *TriggerMsgPriority* oder größer eintrifft.

MQTT_DEPTH
Eine Auslösenachricht wird geschrieben, wenn in der Warteschlange die Anzahl an Nachrichten der Priorität *TriggerMsgPriority* oder größer gleich dem Wert für *TriggerDepth* ist oder diesen übersteigt. Nachdem die Auslösenachricht geschrieben wurde, wird *TriggerControl* auf MQTC_OFF gesetzt, um weitere Auslösevorgänge zu verhindern, bis es wieder explizit aktiviert wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_TRIGGER_TYPE im MQINQ-Aufruf bestimmt. Um den Wert dieses Attributs zu ändern, verwenden Sie den MQSET-Aufruf.

Usage (MQLONG)
Gibt an, für welche Warteschlange die Warteschlange verwendet wird.

Lokal	Modell	Alias	Fern	Cluster
X	X			

Folgende Werte sind möglich:

MQUS_NORMAL
Diese Warteschlange wird von Anwendungen zum Einreihen und Abrufen von Nachrichten verwendet. Die Warteschlange ist keine Übertragungswarteschlange.

MQUS_TRANSMISSION
Diese Warteschlange wird zur Aufnahme von Nachrichten verwendet, die für ferne Warteschlangenmanager bestimmt sind. Wenn eine Anwendung eine Nachricht an eine ferne Warteschlange sendet, speichert der lokale Warteschlangenmanager die Nachricht in einem speziellen Format temporär in der Übertragungswarteschlange. Dann liest der Nachrichtenkanalagent die Nachricht aus der Übertragungswarteschlange und transportiert sie zum fernen Warteschlangenmanager. Weitere Informationen zu Übertragungswarteschlangen finden Sie im Abschnitt Übertragungswarteschlange definieren.

Nur berechtigte Anwendungen können eine Übertragungswarteschlange für MQOO_OUTPUT öffnen, um Nachrichten direkt einzureihen. Normalerweise führen dies nur Dienstprogrammanwendungen aus. Stellen Sie sicher, dass das Nachrichtendatenformat korrekt ist (siehe „MQXQH – Header der Übertragungswarteschlange” auf Seite 609), weil sonst während des Übertragungsprozesses Fehler auftreten können. Kontext wird nicht übergeben oder festgelegt, außer eine der Kontextoptionen MQPMO_*_CONTEXT ist angegeben.

Der Wert dieses Attributs wird mit dem Selektor MQIA_USAGE im MQINQ-Aufruf bestimmt.

XmitQName (MQCHAR48)
Dies ist der Name der Übertragungswarteschlange. Wenn dieses Attribut bei einer Öffnung entweder für eine ferne Warteschlange oder für eine WS-Manager-Aliasdefinition belegt ist, gibt es den Namen der lokalen Übertragungswarteschlange an, die zur Weiterleitung der Nachricht verwendet werden soll.

Lokal	Modell	Alias	Fern	Cluster
			X	

Wenn *XmitQName* leer ist, wird die lokale Warteschlange mit demselben Namen wie *RemoteQMgrName* als Übertragungswarteschlange verwendet. Wenn es keine Warteschlange mit dem Namen *RemoteQMgrName* gibt, wird die Warteschlange verwendet, die durch das Warteschlangenmanagerattribut *DefXmitQName* angegeben ist.

Dieses Attribut wird ignoriert, wenn die Definition als Warteschlangenmanager-Aliasname verwendet wird und *RemoteQMgrName* den Namen des lokalen Warteschlangenmanagers angibt. Es wird auch ignoriert, wenn die Definition als Aliaswarteschlange für Antwortnachrichten verwendet wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_XMIT_Q_NAME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_Q_NAME_LENGTH angegeben.

Attribute für Namenslisten

Die folgende Tabelle enthält eine Zusammenfassung der Attribute, die für Namenslisten spezifisch sind. Die Attribute werden in alphabetischer Reihenfolge beschrieben.

Namenslisten werden von allen WebSphere MQ-Systemen sowie von MQI-Clients von WebSphere MQ, die mit diesen Systemen verbunden sind, unterstützt.

Anmerkung: Die Namen der in diesem Abschnitt erläuterten Attribute sind beschreibende Namen, die mit den MQINQ- und MQSET-Aufrufen verwendet werden; es sind dieselben Namen wie für die PCF-Befehle. Wenn Attribute mit MQSC-Befehlen definiert, geändert oder angezeigt werden, werden alternative Kurznamen verwendet. Weitere Informationen finden Sie unter Scriptbefehle (MQSC).

Tabelle 576. Attribute für Namenslisten

Attribut	Beschreibung
AlterationDate	Datum der letzten Änderung der Definition
AlterationTime	Uhrzeit der letzten Änderung der Definition
NameCount	Anzahl Namen in der Namensliste
NamelistDesc	Namenslistenbeschreibung
NamelistName	Name der Namensliste
Names	Liste mit *NameCount*-Namen
NamelistType	Namenslistentyp
QSGDisp	Disposition der Gruppe mit gemeinsamer Warteschlange

AlterationDate (MQCHAR12)

Dieses Attribut gibt das Datum an, an dem die Definition zuletzt geändert wurde. Das Format des Datums ist YYYY-MM-DDund wird mit zwei abschließenden Leerzeichen aufgefüllt, damit die Länge 12 Byte beträgt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_DATE im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_DATE_LENGTH vorgegeben.

AlterationTime (MQCHAR8)

Dieses Attribut gibt die Uhrzeit an, zu der die Definition zuletzt geändert wurde. Das Zeitformat lautet HH.MM.SS.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_TIME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_TIME_LENGTH vorgegeben.

NameCount (MQLONG)

Dieses Attribut gibt die Anzahl Namen an, die sich derzeit in der Namensliste befinden. Der Wert ist größer-gleich null. Der folgende Wert ist definiert:

MQNC_MAX_NAMELIST_NAME_COUNT
Maximale Anzahl an Namen in einer Namensliste.

Der Wert dieses Attributs wird mit dem Selektor MQIA_NAME_COUNT im MQINQ-Aufruf bestimmt.

NamelistDesc (MQCHAR64)

Verwenden Sie dieses Feld, um eine beschreibende Erläuterung einzugeben. Der Wert wird vom Definitionsprozess erstellt. Der Feldinhalt ist für den Warteschlangenmanager nicht von Bedeutung, es ist aber erforderlich, dass das Feld nur solche Zeichen enthält, die angezeigt werden können. Das Feld darf keine Nullzeichen enthalten; ggf. wird es rechts mit Leerzeichen aufgefüllt. In einer DBCS-Installation kann der Text DBCS-Zeichen enthalten (mit einer maximalen Feldlänge von 64 Byte).

Anmerkung: Wenn das Feld Zeichen enthält, die nicht Bestandteil des Zeichensatzes des Warteschlangenmanagers sind (gemäß der Definition im Warteschlangenmanagerattribut *CodedCharSetId*), werden sie möglicherweise falsch übersetzt, wenn das Feld an einen anderen Warteschlangenmanager gesendet wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_NAMELIST_DESC im MQINQ-Aufruf bestimmt.

Die Länge dieses Attributs wird durch MQ_NAMELIST_DESC_LENGTH angegeben.

NamelistName (MQCHAR48)

Dieses Attribut gibt den Namen einer Namensliste an, die auf dem lokalen Warteschlangenmanager definiert ist. Weitere Informationen zu Namenslistennamen finden Sie im Abschnitt Weitere Objektnamen.

Jede Namensliste hat einen Namen, der sich von den Namen anderer Namenslisten, die zu dem Warteschlangenmanager gehören, unterscheidet, der aber möglicherweise die Namen anderer Warteschlangenmanagerobjekte von verschiedenen Typen dupliziert (z. B. von Warteschlangen).

Der Wert dieses Attributs wird mit dem Selektor MQCA_NAMELIST_NAME im MQINQ-Aufruf bestimmt.

Die Länge dieses Attributs wird durch MQ_NAMELIST_NAME_LENGTH angegeben.

NamelistType (MQLONG)

Dieses Attribut gibt die Spezifik der Namen in der Namensliste an und zeigt an, wie die Namensliste verwendet wird. Folgende Werte sind möglich:

MQNT_NONE
Namensliste ohne zugewiesenen Typ

MQNT_Q
Namensliste mit Namen von Warteschlangen

MQNT_CLUSTER
Namensliste mit Namen von Clustern

MQNT_AUTH_INFO
Namensliste mit Namen von Authentifizierungs-Informationsobjekten

Der Wert dieses Attributs wird mit dem Selektor MQIA_NAMELIST_TYPE im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

Names (MQCHAR48xNameCount)

Dies ist eine Liste der Namen von *NameCount*, wobei jeder Name der Name eines Objekts ist, das für den lokalen Warteschlangenmanager definiert ist. Weitere Informationen zu Objektnamen finden Sie unter Regeln für die Benennung von IBM WebSphere MQ -Objekten .

Der Wert dieses Attributs wird mit dem Selektor MQCA_NAMES im MQINQ-Aufruf bestimmt.

Die Länge jeden Namens in der Liste wird durch MQ_OBJECT_NAME_LENGTH angegeben.

QSGDisp (MQLONG)

Dieses Attribut gibt die Disposition der Namensliste an. Folgende Werte sind möglich:

MQQSGD_Q_MGR
Das Objekt hat Warteschlangenmanager-Disposition: Die Objektdefinition ist nur dem lokalen Warteschlangenmanager bekannt. Anderen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange ist die Definition nicht bekannt.

Jeder Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann ein Objekt mit demselben Namen und Typ wie das aktuelle Objekt haben, aber es handelt sich um separate Objekte ohne Korrelation. Auch die Attribute müssen nicht dieselben sein.

MQQSGD_COPY
Das Objekt ist eine lokale Kopie einer Master-Objektdefinition, die im gemeinsam genutzten Repository existiert. Jeder einzelne Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann über eine eigene Kopie des Objekts verfügen. Anfänglich haben alle Kopien die gleichen Attribute, aber mit MQSC-Befehlen können Sie jede Kopie ändern, wodurch sich die jeweiligen Attribute von denen der anderen Kopien unterscheiden. Die Attribute der Kopien werden resynchronisiert, wenn die Master-Definition im gemeinsam genutzten Repository geändert wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_QSG_DISP im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

Attribute für Prozessdefinitionen

Die folgende Tabelle enthält eine Zusammenfassung der Attribute, die für Prozessdefinitionen spezifisch sind. Die Attribute werden in alphabetischer Reihenfolge beschrieben.

Anmerkung: Die Namen der Attribute in diesem Abschnitt sind beschreibende Namen, die mit den MQINQ- und MQSET-Aufrufen verwendet werden; es sind dieselben Namen wie für die PCF-Befehle. Wenn Attribute mit MQSC-Befehlen definiert, geändert oder angezeigt werden, werden alternative Kurznamen verwendet. Weitere Informationen finden Sie unter Scriptbefehle (MQSC).

Tabelle 577. Attribute für Prozessdefinitionen

Attribut	Beschreibung
AlterationDate	Datum der letzten Änderung der Definition
AlterationTime	Uhrzeit der letzten Änderung der Definition
ApplId	Anwendungs-ID
ApplType	Anwendungstyp
EnvData	Umgebungsdaten
ProcessDesc	Prozessbeschreibung
ProcessName	Prozessname
QSGDisp	Disposition der Gruppe mit gemeinsamer Warteschlange
UserData	Benutzerdaten

AlterationDate (MQCHAR12)

Dieses Attribut gibt das Datum an, an dem die Definition zuletzt geändert wurde. Das Format des Datums ist YYYY-MM-DDund wird mit zwei abschließenden Leerzeichen aufgefüllt, damit die Länge 12 Byte beträgt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_DATE im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_DATE_LENGTH vorgegeben.

AlterationTime (MQCHAR8)

Dieses Attribut gibt die Uhrzeit an, zu der die Definition zuletzt geändert wurde. Das Zeitformat lautet HH.MM.SS.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ALTERATION_TIME im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_TIME_LENGTH vorgegeben.

ApplId (MQCHAR256)

Dieses Attribut ist eine Zeichenfolge, die die Anwendung angibt, die gestartet werden soll. Diese Informationen werden von einer Auslösemonitoranwendung verwendet, die Nachrichten in einer Initialisie-

rungswarteschlange verarbeitet; die Informationen werden als Bestandteil der Auslösenachricht an die Initialisierungswarteschlange gesendet.

Der Inhalt von *ApplId* richtet sich nach der Auslösemonitoranwendung. Der von WebSphere MQ bereitgestellte Auslösemonitor verlangt, dass *ApplId* der Name eines ausführbaren Programms ist. Die folgenden Hinweise gelten für die jeweiligen Umgebungen:

- Unter z/OS muss *ApplId* Folgendes sein:
  - Eine CICS-Transaktions-ID für Anwendungen, die mit der CICS-Auslösemonitortransaktion CKTI gestartet werden
  - Eine IMS-Transaktions-ID für Anwendungen, die mithilfe des IMS-Auslösemonitors CSQQTRMN gestartet werden
- Auf Windows-Systemen kann dem Programmnamen ein Laufwerk und ein Verzeichnispfad vorangestellt werden.
- Auf UNIX-Systemen kann dem Programmnamen ein Verzeichnispfad vorangestellt werden.

Die Zeichenfolge darf keine Nullen enthalten. Sie wird ggf. rechts mit Leerzeichen aufgefüllt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_APPL_ID im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_PROCESS_APPL_ID_LENGTH angegeben.

ApplType (MQLONG)

Dieses Attribut gibt die Spezifik des Programms an, das als Reaktion auf den Empfang einer Auslösenachricht gestartet werden soll. Diese Informationen werden von einer Auslösemonitoranwendung verwendet, die Nachrichten in einer Initialisierungswarteschlange verarbeitet; die Informationen werden als Bestandteil der Auslösenachricht an die Initialisierungswarteschlange gesendet.

ApplType kann jeden Wert haben, aber die folgenden Werte werden für Standardtypen empfohlen; beschränken Sie benutzerdefinierte Anwendungstypen auf Werte im Bereich von MQAT_USER_FIRST bis MQAT_USER_LAST:

MQAT_AIX
AIX-Anwendung (selber Wert wie MQAT_UNIX)

MQAT_BATCH
Stapelanwendung

MQAT_BROKER
Brokeranwendung

MQAT_CICS
CICS-Transaktion

MQAT_CICS_BRIDGE
CICS-Brückenanwendung

MQAT_CICS_VSE
CICS/VSE-Transaktion

MQAT_DOS
WebSphere MQ MQI-Clientanwendung auf PC DOS

MQAT_IMS
IMS-Anwendung

MQAT_IMS_BRIDGE
IMS-Brückenanwendung

MQAT_JAVA
Java-Anwendung

MQAT_MVS
MVS- oder TSO-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_NOTES_AGENT
Lotus Notes Agent-Anwendung

MQAT_NSK
HP Integrity NonStop Server-Anwendung

MQAT_OS390
OS/390-Anwendung (selber Wert wie MQAT_ZOS)

MQAT_OS400
IBM i-Anwendung

MQAT_RRS_BATCH
RRS-Stapelanwendung

MQAT_UNIX
UNIX-Anwendung

MQAT_UNKNOWN
Unbekannter Anwendungstyp

MQAT_USER
Benutzeranwendung

MQAT_VOS
Stratus VOS-Anwendung.

MQAT_WINDOWS
16-Bit-Windows-Anwendung

MQAT_WINDOWS_NT
32-Bit-Windows-Anwendung

MQAT_WLM
z/OS-Workload-Manager-Anwendung

MQAT_XCF
XCF.

MQAT_ZOS
z/OS-Anwendung

MQAT_USER_FIRST
Niedrigster Wert für einen benutzerdefinierten Anwendungstyp

MQAT_USER_LAST
Höchster Wert für einen benutzerdefinierten Anwendungstyp

Der Wert dieses Attributs wird mit dem Selektor MQIA_APPL_TYPE im MQINQ-Aufruf bestimmt.

EnvData (MQCHAR128)

Gibt eine Zeichenfolge mit Informationen zur Umgebung für die Anwendung an, die gestartet werden soll. Diese Informationen werden von einer Auslösemonitoranwendung verwendet, die Nachrichten in einer Initialisierungswarteschlange verarbeitet; die Informationen werden als Bestandteil der Auslösenachricht an die Initialisierungswarteschlange gesendet.

Der Inhalt von *EnvData* richtet sich nach der Auslösemonitoranwendung. Der Auslösemonitor von WebSphere MQ hängt *EnvData* an die Parameterliste an, die an die gestartete Anwendung übergeben wird. Diese Parameterliste besteht aus der MQTMC2-Struktur, gefolgt von einem Leerzeichen, auf das wiederum *EnvData* folgt. Alle nachfolgenden Leerzeichen werden gelöscht. Die folgenden Hinweise gelten für die jeweiligen Umgebungen:

- Unter z/OS:
  - *EnvData* wird nicht von den in WebSphere MQ verfügbaren Auslösemonitoranwendungen verwendet.
  - Wenn ApplType MQAT_WLM ist, können Sie Standardwerte in EnvData für die Felder "ServiceName" und "ServiceStep" im Arbeitsinformationsheader (MQWIH) angeben.

- Auf UNIX-Systemen kann für *EnvData* das &-Zeichen angegeben werden, wenn die gestartete Anwendung im Hintergrund ausgeführt werden soll.

Die Zeichenfolge darf keine Nullen enthalten. Sie wird ggf. rechts mit Leerzeichen aufgefüllt.

Der Wert dieses Attributs wird mit dem Selektor MQCA_ENV_DATA im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_PROCESS_ENV_DATA_LENGTH angegeben.

ProcessDesc (MQCHAR64)

Verwenden Sie dieses Feld, um eine beschreibende Erläuterung einzugeben. Der Feldinhalt ist für den Warteschlangenmanager nicht von Bedeutung, es ist aber erforderlich, dass das Feld nur solche Zeichen enthält, die angezeigt werden können. Das Feld darf keine Nullzeichen enthalten; ggf. wird es rechts mit Leerzeichen aufgefüllt. In einer DBCS-Installation kann das Feld DBCS-Zeichen enthalten (die maximale Länge beträgt 64 Byte).

Anmerkung: Wenn das Feld Zeichen enthält, die nicht Bestandteil des Zeichensatzes des Warteschlangenmanagers sind (gemäß der Definition im Warteschlangenmanagerattribut *CodedCharSetId*), werden sie möglicherweise falsch übersetzt, wenn das Feld an einen anderen Warteschlangenmanager gesendet wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_PROCESS_DESC im MQINQ-Aufruf bestimmt.

Die Länge dieses Attributs wird durch MQ_PROCESS_DESC_LENGTH angegeben.

ProcessName (MQCHAR48)

Dieses Attribut gibt den Namen einer Prozessdefinition an, die auf dem lokalen Warteschlangenmanager definiert ist.

Jede Prozessdefinition hat einen Namen, der sich von den Namen anderer Prozessdefinitionen, die zu dem Warteschlangenmanager gehören, unterscheidet. Der Name der Prozessdefinition kann aber dem Namen von Warteschlangenmanagerobjekten anderen Typs (z. B. Warteschlangen) entsprechen.

Der Wert dieses Attributs wird mit dem Selektor MQCA_PROCESS_NAME im MQINQ-Aufruf bestimmt.

Die Länge dieses Attributs wird durch MQ_PROCESS_NAME_LENGTH angegeben.

QSGDisp (MQLONG)

Dieses Attribut gibt die Disposition der Prozessdefinition an. Folgende Werte sind möglich:

MQQSGD_Q_MGR
Das Objekt hat Warteschlangenmanager-Disposition: Die Objektdefinition ist nur dem lokalen Warteschlangenmanager bekannt. Anderen Warteschlangenmanagern in der Gruppe mit gemeinsamer Warteschlange ist die Definition nicht bekannt.

Jeder Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann ein Objekt mit demselben Namen und Typ wie das aktuelle Objekt haben, aber es handelt sich um separate Objekte ohne Korrelation. Auch die Attribute müssen nicht dieselben sein.

MQQSGD_COPY
Das Objekt ist eine lokale Kopie einer Master-Objektdefinition, die im gemeinsam genutzten Repository existiert. Jeder einzelne Warteschlangenmanager in der Gruppe mit gemeinsamer Warteschlange kann über eine eigene Kopie des Objekts verfügen. Anfänglich haben alle Kopien die gleichen Attribute, aber mit MQSC-Befehlen können Sie jede Kopie ändern, wodurch sich die jeweiligen Attribute von denen der anderen Kopien unterscheiden. Die Attribute der Kopien werden resynchronisiert, wenn die Master-Definition im gemeinsam genutzten Repository geändert wird.

Der Wert dieses Attributs wird mit dem Selektor MQIA_QSG_DISP im MQINQ-Aufruf bestimmt.

Dieses Attribut wird nur unter z/OS unterstützt.

UserData (MQCHAR128)

Das Attribut UserData gibt eine Zeichenfolge mit Benutzerdaten für die Anwendung an, die gestartet werden soll. Diese Informationen werden von einer Auslösemonitoranwendung verwendet, die Nachrichten in der Initialisierungswarteschlange verarbeitet, oder von der Anwendung, die vom Auslösemonitor gestartet wird. Die Informationen werden als Bestandteil der Auslösenachricht an die Initialisierungswarteschlange gesendet.

Der Inhalt von *UserData* richtet sich nach der Auslösemonitoranwendung. Der Auslösemonitor, der von WebSphere MQ bereitgestellt wird, übergibt *UserData* als Teil der Parameterliste an die gestartete Anwendung. Die Parameterliste besteht aus der MQTMC2-Struktur (mit *UserData*), gefolgt von einem Leerzeichen, auf das wiederum *EnvData* folgt; alle nachfolgenden Leerzeichen werden gelöscht.

Die Zeichenfolge darf keine Nullen enthalten. Sie wird ggf. rechts mit Leerzeichen aufgefüllt. Unter Microsoft Windows darf die Zeichenfolge keine Anführungszeichen enthalten, wenn die Prozessdefinition an **runmqtrm** übergeben wird.

Der Wert dieses Attributs wird mit dem Selektor MQCA_USER_DATA im MQINQ-Aufruf bestimmt. Die Länge dieses Attributs wird durch MQ_PROCESS_USER_DATA_LENGTH angegeben.

Rückgabecodes

Für jeden WebSphere MQ Message Queue Interface (MQI)- und WebSphere MQ Administration Interface (MQAI)-Aufruf werden vom Warteschlangenmanager oder von einer Exitroutine ein **Beendigungscode** und ein **Ursachencode** zurückgegeben, um den Erfolg oder Fehlschlag des Aufrufs anzuzeigen.

Anwendungen dürfen nicht davon abhängig sein, dass Fehler in einer bestimmten Reihenfolge überprüft werden, es sei denn, dies ist ausdrücklich vermerkt. Wenn durch einen Aufruf mehr als ein Beendigungscode oder Ursachencode erzeugt werden kann, hängt es von der Implementierung ab, welcher dieser Fehler zurückgemeldet wird.

Anwendungen, die nach einem WebSphere MQ-API-Aufruf eine Überprüfung auf eine erfolgreiche Beendigung durchführen, müssen immer den Beendigungscode überprüfen. Auf keinen Fall darf der Wert des Beendigungscodes auf der Basis des Werts des Ursachencodes vorausgesetzt werden.

Beendigungscodes

Der Beendigungscodeparameter (*CompCode*) ermöglicht es der aufrufenden Anwendung, schnell zu überprüfen, ob der Aufruf erfolgreich oder teilweise ausgeführt wurde oder ob er fehlgeschlagen ist.In der folgenden Liste sind die Beendigungscodes detaillierter als in den Aufrufbeschreibungen aufgeführt:

MQCC_OK
Der Aufruf wurde vollständig ausgeführt; alle Ausgabeparameter wurden gesetzt. Der Parameter *Reason* hat in diesem Fall immer den Wert MQRC_NONE.

MQCC_WARNING
Der Aufruf wurde teilweise ausgeführt. Möglicherweise wurden zusätzlich zu den Ausgabeparametern *CompCode* und *Reason* weitere Ausgabeparameter gesetzt. Der Parameter *Reason* liefert zusätzliche Informationen zur teilweisen Ausführung.

MQCC_FAILED
Die Verarbeitung des Aufrufs wurde nicht beendet. Der Status des Warteschlangenmanagers ist unverändert; andernfalls wird gesondert darauf hingewiesen. Die Ausgabeparameter *CompCode* und *Reason* wurden gesetzt. Sonstige Parameter sind unverändert; andernfalls wird darauf hingewiesen.

Die Ursache kann ein Fehler im Anwendungsprogramm oder das Ergebnis einer bestimmten Situation außerhalb des Programms sein, z. B. wenn dem Benutzer die Berechtigung entzogen wurde. Der Parameter *Reason* liefert zusätzliche Informationen zu dem Fehler.

Ursachencodes

Der Ursachencodeparameter (*Reason*) dient zur Qualifikation des Beendigungscodeparameters (*CompCode*).

Wenn es keine besondere Ursache zurückzumelden gibt, wird MQRC_NONE zurückgegeben. Ein erfolgreicher Aufruf gibt MQCC_OK und MQRC_NONE zurück.

Wenn der Beendigungscode entweder MQCC_WARNING oder MQCC_FAILED lautet, gibt der Warteschlangenmanager immer eine qualifizierende Ursache zurück; Details finden Sie in der Aufrufbeschreibung.

Wenn Benutzerexitroutinen Beendigungscodes und Ursachen angeben, müssen sie diesen Regeln entsprechen. Darüber hinaus müssen spezielle Ursachenwerte, die von Benutzerexits definiert werden, kleiner als 0 sein, um sicherzustellen, dass keine Konflikte mit Werten entstehen, die vom Warteschlangenmanager definiert werden. Wo dies geeignet ist, können Exits Ursachen angeben, die bereits vom Warteschlangenmanager definiert sind.

Ursachencodes werden des Weiteren auch an den folgenden Orten angegeben:

- im Feld *Reason* der MQDLH-Struktur und
- im Feld *Feedback* der MQMD-Struktur.

Vollständige Beschreibungen der Ursachencodes finden Sie unter Ursachencodes .

Regeln zur Überprüfung von MQI-Optionen

In diesem Abschnitt werden die Situationen aufgelistet, in denen der Ursachencode MQRC_OPTIONS_ERROR aus einem MQOPEN-, MQPUT-, MQPUT1-, MQGET-, MQCLOSE-oder MQSUB-Aufruf erstellt wird.

MQOPEN-Aufruf

Für die Optionen des MQOPEN-Aufrufs:

- Es muss mindestens *eine* der folgenden Optionen angegeben werden:
  - 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]
- Es ist nur *eine* der folgenden Optionen zulässig:
  - MQOO_READ_AHEAD
  - MQOO_NO_READ_AHEAD
  - MQOO_READ_AHEAD_AS_Q_DEF

1. Es ist nur *eine* der folgenden Optionen zulässig:
   - MQOO_INPUT_EXCLUSIVE
   - MQOO_INPUT_SHARED
   - MQOO_INPUT_AS_Q_DEF
2. Es ist nur *eine* der folgenden Optionen zulässig:
   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NOT_FIXED
   - MQOO_BIND_ON_GROUP

- MQOO_BIND_AS_Q_DEF

  Anmerkung: Die oben aufgelisteten Optionen schließen sich gegenseitig aus. Da der Wert von MQOO_BIND_AS_Q_DEF 0 ist, führt die Angabe dieser Option zusammen mit einer der beiden Bindeoptionen jedoch nicht zu dem Ursachencode MQRC_OPTIONS_ERROR. MQOO_BIND_AS_Q_DEF wird zur Unterstützung der Programmdokumentation bereitgestellt.

- Wenn MQOO_SAVE_ALL_CONTEXT angegeben wird, muss auch eine der MQOO_INPUT_*-Optionen angegeben werden.
- Wenn die Option MQOO_SET_*_CONTEXT oder MQOO_PASS_*_CONTEXT angegeben wird, muss auch MQOO_OUTPUT angegeben werden.
- Wenn MQOO_CO_OP angegeben wird, muss auch MQOO_BROWSE angegeben werden.
- Wenn MQOO_NO_MULTICAST angegeben ist, muss auch MQOO_OUTPUT angegeben werden.

MQPUT, Aufruf

Für die Optionen zum Einreihen von Nachrichten:

- Die Kombination von MQPMO_SYNCPOINT und MQPMO_NO_SYNCPOINT ist nicht zulässig.
- Es ist nur *eine* der folgenden Optionen zulässig:
  - MQPMO_DEFAULT_CONTEXT
  - MQPMO_NO_CONTEXT
  - MQPMO_PASS_ALL_CONTEXT
  - MQPMO_PASS_IDENTITY_CONTEXT
  - MQPMO_SET_ALL_CONTEXT
  - MQPMO_SET_IDENTITY_CONTEXT
- Es ist nur *eine* der folgenden Optionen zulässig:
  - MQPMO_ASYNC_RESPONSE
  - MQPMO_SYNC_RESPONSE
  - MQPMO_RESPONSE_AS_TOPIC_DEF
  - MQPMO_RESPONSE_AS_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY ist nicht zulässig (sie ist nur im Aufruf MQPUT1 gültig).

MQPUT1, Aufruf

Für die Optionen zum Einreihen von Nachrichten gelten dieselben Regeln wie für den MQPUT-Aufruf, mit folgenden Ausnahmen:

- MQPMO_ALTERNATE_USER_AUTHORITY ist zulässig.
- MQPMO_LOGICAL_ORDER ist *nicht* zulässig.

MQGET-Aufruf

Für die Optionen zum Abrufen von Nachrichten:

- Es ist nur *eine* der folgenden Optionen zulässig:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_SYNCPOINT
  - MQGMO_SYNCPOINT_IF _PERSISTENT
- Es ist nur *eine* der folgenden Optionen zulässig:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR

  - MQGMO_BROWSE_NEXT
  - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT ist zusammen mit einer der folgenden Optionen nicht zulässig:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_LOCK
  - MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT ist zusammen mit einer der folgenden Optionen nicht zulässig:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_COMPLETE_MSG
  - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT erfordert die Angabe von MQGMO_SYNCPOINT.
- Die Kombination von MQGMO_WAIT und MQGMO_SET_SIGNAL ist nicht zulässig.
- Wenn MQGMO_LOCK angegeben wird, muss auch eine der folgenden Optionen angegeben werden:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
- Wenn MQGMO_UNLOCK angegeben wird, sind nur folgende Optionen zulässig:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_NO_WAIT

MQCLOSE, Aufruf

Für die Optionen des MQCLOSE-Aufrufs:

- Die Kombination von MQCO_DELETE und MQCO_DELETE_PURGE ist nicht zulässig.
- Es ist nur eine der folgenden Optionen zulässig:
  - MQCO_KEEP_SUB
  - MQCO_REMOVE_SUB

MQSUB, Aufruf

Für die Optionen des MQSUB-Aufrufs:

- Es muss mindestens eine der folgenden Optionen angegeben werden:
  - MQSO_ALTER
  - MQSO_RESUME
  - MQSO_CREATE
- Es ist nur eine der folgenden Optionen zulässig:
  - MQSO_DURABLE
  - MQSO_NON_DURABLE

  Anmerkung: Die oben aufgelisteten Optionen schließen sich gegenseitig aus. Da der Wert von MQSO_NON_DURABLE 0 ist, führt die Angabe dieser Option zusammen mit MQSO_DURABLE jedoch

nicht zu dem Ursachencode MQRC_OPTIONS_ERROR. MQSO_NON_DURABLE wird zur Unterstützung der Programmdokumentation bereitgestellt.

- Die Kombination von MQSO_GROUP_SUB und MQSO_MANAGED ist nicht zulässig.
- MQSO_GROUP_SUB erfordert die Angabe von MQSO_SET_CORREL_ID.
- Es ist nur eine der folgenden Optionen zulässig:
  - MQSO_ANY_USERID
  - MQSO_FIXED_USERID
- MQSO_NEW_PUBLICATIONS_ONLY ist nur in Kombination mit MQSO_CREATE zulässig.
- Die Kombination von MQSO_PUBLICATIONS_ON_REQUEST und SubLevel größer als 1 ist nicht zulässig.
- Es ist nur eine der folgenden Optionen zulässig:
  - MQSO_WILDCARD_CHAR
  - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST erfordert die Angabe von MQSO_MANAGED.

Befehlsnachrichten für eingereihtes Publish/Subscribe

Eine Anwendung kann mit MQRFH2-Befehlsnachrichten eine eingereihte Publish/Subscribe-Anwendung steuern.

Eine Anwendung, die MQRFH2 für Publish/Subscribe verwendet, kann die folgenden Befehlsnachrichten an die Warteschlange SYSTEM.BROKER.CONTROL.QUEUE senden:

- „Nachricht zum Löschen von Veröffentlichungen” auf Seite 878
- „Nachricht 'Deregister Subscriber'” auf Seite 879
- „Nachricht veröffentlichen” auf Seite 883
- „Nachricht 'Register Subscriber'” auf Seite 886
- „Nachricht 'Request Update'” auf Seite 891

Beim Schreiben von eingereihten Publish/Subscribe-Anwendungen benötigen Sie Kenntnisse über die Nachrichten, die Antwortnachricht des Warteschlangenmanagers und den Nachrichtendeskriptor MQMD). Weitere Informationen finden Sie in den folgenden Abschnitten:

- „Nachricht 'Queue Manager Response'” auf Seite 893
- „MQMD-Einstellungen für Veröffentlichungen, die von einem Warteschlangenmanager weitergeleitet wurden” auf Seite 899
- „MQMD-Einstellungen in Antwortnachrichten des Warteschlangenmanagers” auf Seite 900
- „Publish/Subscribe-Ursachencodes” auf Seite 895

Die Befehle befinden sich in einem <psc>-Ordner im Feld **NameValueData** des MQRFH2-Headers. Die Nachricht, die von einem Broker als Antwort auf eine Befehlsnachricht gesendet werden kann, ist in einem <pscr>-Ordner enthalten.

In den Beschreibungen zu jedem Befehl werden die Eigenschaften aufgeführt, die in einem Ordner enthalten sein können. Sofern nichts anderes angegeben ist, sind die Eigenschaften optional und können nur einmal vorkommen.

Die Namen der Eigenschaften werden als <Command> angezeigt.
Werte müssen im Zeichenfolgeformat angegeben werden; Beispiel: Publish.
Eine Zeichenfolgekonstante, die den Wert einer Eigenschaft darstellt, wird in runden Klammern angezeigt; Beispiel: (MQPSC_PUBLISH).
Zeichenfolgekonstanten werden in der Headerdatei cmqpsc.h definiert, die mit dem Warteschlangenmanager geliefert wird.

Nachricht zum Löschen von Veröffentlichungen

Die Befehlsnachricht für **Delete Publication** wird von einer Veröffentlichungskomponente oder einem anderen Warteschlangenmanager an einen Warteschlangenmanager gesendet; sie weist den Warteschlangenmanager an, alle ständigen Veröffentlichungen zu den angegebenen Themen zu löschen.

Diese Nachricht wird an eine Warteschlange gesendet, die von der eingereihten Publish/Subscribe-Schnittstelle des Warteschlangenmanagers überwacht wird.

Die Eingabewarteschlange sollte die Warteschlange sein, an die die ursprüngliche Veröffentlichung gesendet wurde.

Wenn Sie die Berechtigung für einige, aber nicht alle Themen haben, die in der **Delete Publication** -Befehlsnachricht angegeben sind, werden nur diese Themen gelöscht. Eine Nachricht **Broker Response** (Brokerantwort) gibt an, welche Themen nicht gelöscht werden.

Wenn ein **Publish**-Befehl mehrere Themen enthält, werden von dem Befehl **Delete Publication**, der mit einigen, aber nicht allen dieser Themen übereinstimmt, entsprechend nur die Veröffentlichungen für die Themen gelöscht, die im Befehl **Delete Publication** angegeben sind.

Im Abschnitt „MQMD-Einstellungen für Veröffentlichungen, die von einem Warteschlangenmanager weitergeleitet wurden” auf Seite 899 finden Sie weitere Informationen zu den Parametern des Nachrichtendeskriptors (MQMD), die zum Senden einer Befehlsnachricht an den Warteschlangenmanager erforderlich sind.

Eigenschaften

< Befehl> (*MQPSC_COMMAND*)
Der Wert ist DeletePub(*MQPSC_DELETE_PUBLICATION*).

Diese Eigenschaft muss angegeben werden.

< Topic> (*MQPSC_TOPIC*)
Bei dem Wert handelt es sich um eine Zeichenfolge mit einem Thema, für das ständige Veröffentlichungen gelöscht werden sollen. Die Zeichenfolge kann Platzhalter enthalten, um Veröffentlichungen zu mehreren Themen zu löschen.

Diese Eigenschaft muss angegeben werden; sie kann für beliebig viele Themen wiederholt werden.

<DelOpt> (*MQPSC_DELETE_OPTION*)
Für die Eigenschaft zum Löschen von Optionen kann einer der folgenden Werte angegeben werden:

Local (*MQPSC_LOCAL*)

Alle ständigen Veröffentlichungen für die angegebenen Themen werden im lokalen Warteschlangenmanager (dem Warteschlangenmanager, an den die Nachricht gesendet wurde) gelöscht, unabhängig davon, ob diese Nachrichten mit der Option Local veröffentlicht wurden oder nicht.

Veröffentlichungen in anderen Warteschlangenmanagern sind hiervon nicht betroffen.

None (*MQPSC_NONE*)

Für alle Optionen werden die Standardwerte verwendet. Dies entspricht dem Übergehen der Eigenschaft DelOpt. Wenn gleichzeitig andere Optionen angegeben werden, wird die Option None ignoriert.

Wird diese Eigenschaft ausgelassen, werden standardmäßig alle ständigen Veröffentlichungen zu den angegebenen Themen in allen Warteschlangenmanagern im Netz gelöscht, unabhängig davon, ob diese Nachrichten mit der Option Local veröffentlicht wurden oder nicht.

Beispiel

Hier ein Beispiel für `NameValueData` für die Befehlsnachricht **`Delete Publication`**. Es wird von der Musteranwendung verwendet, um die ständige Veröffentlichung mit dem aktuellen Spielstand zwischen Team1 und Team2 im lokalen Warteschlangenmanager zu löschen.

```
<psc>
 <Command>DeletePub</Command>
 <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
 <DelOpt>Local</DelOpt>
</psc>
```

Nachricht 'Deregister Subscriber'

Die Befehlsnachricht für **`Deregister Subscriber`** wird von einem Subskribenten oder einer anderen Anwendung im Auftrag eines Subskribenten an einen Warteschlangenmanager gesendet, um anzuzeigen, dass keine Nachrichten mehr empfangen werden sollen, die mit den angegebenen Parametern übereinstimmen.

Diese Nachricht wird an die Steuerwarteschlange SYSTEM.BROKER.CONTROL.QUEUE des Warteschlangenmanagers gesendet. Der Benutzer muss über die entsprechende Berechtigung zum Einreihen einer Nachricht in diese Warteschlange verfügen.

Im Abschnitt MQMD-Einstellungen für Veröffentlichungen, die von einem Warteschlangenmanager weitergeleitet wurden finden Sie weitere Informationen zu den Parametern des Nachrichtendeskriptors (MQMD), die zum Senden einer Befehlsnachricht an den Warteschlangenmanager erforderlich sind.

Die Registrierung einer einzelnen Subskription kann zurückgenommen werden, indem das zugehörige Thema, der Subskriptionspunkt und die Filterwerte der ursprünglichen Subskription angegeben werden. Wenn einer dieser Werte in der ursprünglichen Subskription nicht angegeben wurde (d. h. wenn die Standardwerte übernommen wurden), sollte er bei der Zurücknahme der Registrierung übergangen werden.

Mit der Option `DeregAll` kann die Registrierung für alle Subskriptionen eines Subskribenten oder einer Gruppe von Subskribenten zurückgenommen werden. Wenn beispielsweise die Option `DeregAll` mit einem Subskriptionspunkt angegeben ist (aber ohne Thema oder Filter), wird die Registrierung aller Subskriptionen für den Subskribenten im angegebenen Subskriptionspunkt zurückgenommen, unabhängig von Thema und Filter. Es ist jede beliebige Kombination aus Thema, Filter und Subskriptionspunkt zulässig; wenn alle drei Elemente angegeben sind, kann nur eine Subskription übereinstimmen und die Option `DeregAll` wird ignoriert.

Die Nachricht muss von dem Subskribenten gesendet werden, der die Subskription registriert hat; dies wird durch die Prüfung der Benutzer-ID des Subskribenten bestätigt.

Die Registrierung von Subskriptionen kann auch von einem Systemadministrator mithilfe von MQSC- oder PCF-Befehlen zurückgenommen werden. Die Subskriptionen, die mit einer temporären dynamischen Warteschlange registriert wurden, werden jedoch der Warteschlange, nicht nur dem Warteschlangennamen zugeordnet. Wenn die Warteschlange gelöscht wird (entweder explizit oder durch die Trennung der Anwendung vom Warteschlangenmanager), kann die Registrierung der Subskriptionen für diese Warteschlange nicht mehr mit dem Befehl **`Deregister Subscriber`** zurückgenommen werden. Die Registrierung der Subskriptionen kann über die Developer Workbench zurückgenommen werden. Der Warteschlangenmanager entfernt die Subskriptionen dann automatisch bei der nächsten Übereinstimmung einer Veröffentlichung mit der Subskription oder beim nächsten Neustart des Warteschlangenmanagers. Normalerweise sollten Anwendungen die Registrierung ihrer Subskriptionen vor dem Löschen der Warteschlange oder vor dem Trennen der Verbindung zum Warteschlangenmanager zurücknehmen.

Wenn ein Subskribent eine Nachricht zur Rücknahme der Registrierung einer Subskription sendet und eine Antwortnachricht mit der Mitteilung empfängt, dass die Bearbeitung erfolgreich war, greifen einige Veröffentlichungen möglicherweise weiterhin auf die Warteschlange für Subskribenten zu, wenn sie vom Warteschlangenmanager während der Rücknahme der Registrierung einer Subskription bearbeitet wurden. Wenn die Nachrichten nicht aus der Warteschlange entfernt werden, kann es zu einem Rückstau an unverarbeiteten Nachrichten in der Warteschlange für Subskribenten kommen. Wenn die Anwendung nach einer Zeit im Ruhemodus eine Schleife durchläuft, die einen MQGET-Aufruf mit dem zugehörigen Parameter `CorrelId` enthält, werden diese Nachrichten aus der Warteschlange entfernt.

Entsprechend ist die Warteschlange möglicherweise nicht leer, wenn der Subskribent eine permanente dynamische Warteschlange verwendet, die Registrierung der Warteschlange zurücknimmt und die Warteschlange mit der Option *MQCO_DELETE_PURGE* in einem MQCLOSE-Aufruf schließt. Falls Veröffentlichungen aus dem Warteschlangenmanager beim Löschen der Warteschlange noch nicht festgeschrieben wurden, wird durch den MQCLOSE-Aufruf der Rückkehrcode MQRC_Q_NOT_EMPTY ausgegeben. Die Anwendung kann dieses Problem vermeiden, indem sie in den Ruhemodus versetzt wird und den MQCLOSE-Aufruf ab und zu erneut ausgibt.

Eigenschaften

< Befehl> (*MQPSC_COMMAND*)
Der Wert ist DeregSub (*MQPSC_DEREGISTER_SUBSCRIBER*).

Diese Eigenschaft muss angegeben werden.

< Topic> (*MQPSC_TOPIC*)
Der Wert ist eine Zeichenfolge mit dem Thema, dessen Registrierung zurückgenommen werden soll.

Diese Eigenschaft kann optional wiederholt werden, wenn die Registrierung mehrerer Themen zurückgenommen werden soll. Sie kann weggelassen werden, wenn DeregAll in <RegOpt> angegeben ist.

Bei den angegebenen Themen kann es sich um eine Untergruppe der registrierten Themen handeln, wenn der Subskribent die Subskriptionen anderer Themen beibehalten möchte. Es können Platzhalterzeichen verwendet werden, aber eine Themenzeichenfolge mit Platzhalterzeichen muss exakt mit der entsprechenden Zeichenfolge übereinstimmen, die in der Befehlsnachricht für **Deregister Subscriber** angegeben wurde.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
Der Wert ist eine Zeichenfolge, die den Subskriptionspunkt angibt, aus dem die Subskription freigegeben werden soll.

Die Eigenschaft darf nicht wiederholt werden. Sie kann weggelassen werden, wenn ein <Topic> angegeben ist oder wenn DeregAll in <RegOpt> angegeben ist. Wenn Sie diese Eigenschaft weglassen, geschieht Folgendes:

- Wenn Sie DeregAll **nicht** angeben, werden Abonnements, die mit der Eigenschaft <Topic> übereinstimmen (und die <Filter> Eigenschaft, falls vorhanden), vom Standard-Abonnementpunkt deregistriert.
- Wenn Sie DeregAll angeben, werden alle Abonnements (die mit den <Topic>- und <Filter>-Eigenschaften übereinstimmen, falls vorhanden) von allen Abonnementspunkten deregistriert.

Beachten Sie, dass Sie den standardmäßigen Subskriptionspunkt nicht explizit angeben können. Deshalb kann nicht die Registrierung aller Subskriptionen nur aus diesem Subskriptionspunkt zurückgenommen werden; Sie müssen die Themen angeben.

<SubIdentity> (*MQPSC_SUBSCRIPTION_IDENTITY*)
Dies ist eine Zeichenfolge variabler Länge mit einer maximalen Länge von 64 Zeichen. Sie dient zur Darstellung einer Anwendung, die Interesse an einer Subskription hat. Der Warteschlangenmanager verwaltet für jede Subskription einen Satz an Subskribentenidentitäten. Jede Subskription kann entweder nur eine einzelne Identität oder eine unbegrenzte Anzahl von Identitäten enthalten.

Wenn sich die Eigenschaft SubIdentity im Identitätssatz für die Subskription befindet, wird sie aus dem Satz entfernt. Wenn sich daraus ein leerer Identitätssatz ergibt, wird die Subskription aus dem Warteschlangenmanager entfernt, es sei denn, die Option LeaveOnly ist als Wert der Eigenschaft RegOpt angegeben. Wenn der Identitätssatz noch weitere Identitäten enthält, wird die Subskription nicht aus dem Warteschlangenmanager entfernt und der Veröffentlichungsablauf wird nicht unterbrochen.

Wenn die Eigenschaft SubIdentity angegeben ist, der Identitätssatz für die Subskription aber die Eigenschaft SubIdentity nicht enthält, schlägt der Befehl **Deregister Subscriber** mit dem Rückkehrcode *MQRCCF_SUB_IDENTITY_ERROR* fehl.

< Filter > (*MQPSC_FILTER*)
Der Wert ist eine Zeichenfolge, die den Filter angibt, dessen Registrierung zurückgenommen werden soll. Er muss exakt (mit Groß-/Kleinschreibung und allen Leerzeichen) mit einem Subskriptionsfilter übereinstimmen, der zuvor registriert wurde.

Diese Eigenschaft kann optional wiederholt werden, wenn die Registrierung mehrerer Filter zurückgenommen werden soll. Sie kann weggelassen werden, wenn ein <Topic> angegeben ist oder wenn DeregAll in <RegOpt> angegeben ist.

Bei den angegebenen Filtern kann es sich um eine Untergruppe der registrierten Filter handeln, wenn der Subskribent die Subskriptionen für andere Filter beibehalten möchte.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
Die Eigenschaft zum Registrieren von Optionen kann einen der folgenden Werte haben:

DeregAll
(*MQPSC_DEREGISTER_ALL*)

Die Registrierung aller für diesen Subskribenten registrierten übereinstimmenden Subskriptionen wird zurückgenommen.

Wenn DeregAll angegeben ist:

- <Topic>, <SubPoint> und <Filter> können übergangen werden.
- <Topic> und <Filter> können, falls erforderlich, wiederholt werden.
- <SubPoint> darf nicht wiederholt werden.

Wenn DeregAll **nicht** angegeben ist:

- <Topic> muss angegeben werden und kann bei Bedarf wiederholt werden.
- <SubPoint> und <Filter> können übergangen werden.
- <SubPoint> darf nicht wiederholt werden.
- <Filter> kann, falls erforderlich, wiederholt werden.

Wenn sowohl 'Topic' als auch 'Filter' wiederholt werden, werden alle Subskriptionen entfernt, die mit einer der möglichen Kombinationen dieser beiden Eigenschaften übereinstimmen. Beispiel: Ein **Deregister Subscriber** -Befehl, der drei Themen und drei Filter angibt, versucht, neun Subskriptionen zu entfernen.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Die im Feld CorrelId des Nachrichtendeskriptors (MQMD) angegebene Korrelations-ID (die nicht null sein darf) wird zur Ermittlung des Subskribenten verwendet. Sie muss mit der Korrelations-ID im Feld CorrelId übereinstimmen, die in der ursprünglichen Subskription verwendet wurde.

FullResp
(*MQPSC_FULL_RESPONSE*)

Wenn FullResp angegeben ist, werden in der Antwortnachricht alle Attribute der Subskription zurückgegeben, sofern der Befehl nicht fehlschlägt.

Wenn FullResp angegeben wird, ist DeregAll im Befehl **Deregister Subscriber** nicht zulässig. Es können auch nicht mehrere Themen angegeben werden. Der Befehl schlägt in beiden Fällen mit dem Rückkehrcode *MQRCCF_REG_OPTIONS_ERROR* fehl.

LeaveOnly
(*MQPSC_LEAVE_ONLY*)

Wenn Sie diese Option mit der Eigenschaft SubIdentity angeben, die sich im Identitätssatz für die Subskription befindet, wird die Eigenschaft SubIdentity aus dem Identitätssatz für die Subskription entfernt. Die Subskription wird nicht aus dem Warteschlangenmanager entfernt, selbst wenn der sich daraus ergebende Identitätssatz leer ist. Wenn sich der Wert SubIdentity

nicht im Identitätssatz befindet, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_SUB_IDENTITY_ERROR* fehl.

Wenn der Wert `LeaveOnly` ohne Eigenschaft `SubIdentity` angegeben wird, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_REG_OPTIONS_ERROR* fehl.

Wenn weder `LeaveOnly` noch `SubIdentity` angegeben werden, wird die Subskription unabhängig vom Inhalt des Identitätssatzes für die Subskription entfernt.

None
(MQPSC_NONE)

Für alle Optionen werden die Standardwerte verwendet. Dies entspricht dem Übergehen der Eigenschaft zum Registrieren von Optionen. Wenn gleichzeitig andere Optionen angegeben werden, wird die Option None ignoriert.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Wenn dieser Wert angegeben ist, wird die Identität des Subskribenten (Warteschlange, Warteschlangenmanager und Korrelations-ID) nicht auf eine einzige Benutzer-ID eingeschränkt. Dies unterscheidet sich vom bestehenden Verhalten des Warteschlangenmanagers, der die Benutzer-ID der ursprünglichen Registrierungsnachricht der Identität des Subskribenten zuordnet und anschließend verhindert, dass andere Benutzer diese Identität verwenden. Wenn ein neuer Subskribent die gleiche Identität verwenden möchte, wird der Rückkehrcode *MQRCCF_DUPLICATE_SUBSCRIPTION* zurückgegeben.

Jeder Benutzer, der über die entsprechende Berechtigung verfügt, kann die Subskription ändern oder die Registrierung der Subskription zurücknehmen und so die vorhandene Prüfung vermeiden, in der die Benutzer-ID der Benutzer-ID des ursprünglichen Subskribenten entsprechen muss.

Um diese Option einer vorhandenen Subskription hinzuzufügen, muss der Befehl mit der gleichen Benutzer-ID ausgegeben werden wie die ursprüngliche Subskription.

Wenn für die Subskription, deren Registrierung zurückgenommen werden soll, der Wert `VariableUserId` festgelegt ist, muss dieser Wert auch bei der Rücknahme der Registrierung angegeben werden, um anzuzeigen, für welche Subskription die Registrierung zurückgenommen werden soll. Andernfalls wird die Benutzer-ID des Befehls **Deregister Subscriber** zum Identifizieren der Subskription verwendet. Diese ID wird zusammen mit den anderen Subskribenten-IDs überschrieben, falls ein Subskriptionsname bereitgestellt wird.

Wenn diese Eigenschaft übergangen wird, werden standardmäßig keine Registrierungsoptionen festgelegt.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
Der Wert ist der Name des Warteschlangenmanagers für die Warteschlange des Subskribenten. Er muss mit der Option `QMgrName` übereinstimmen, die in der ursprünglichen Subskription verwendet wurde.

Wird diese Eigenschaft übergangen, wird standardmäßig der Name `ReplyToQMgr` im Nachrichtendeskriptor (MQMD) verwendet. Ist dieser Name leer, wird standardmäßig der Name des Warteschlangenmanagers übernommen.

< Warteschlangenname> (*MQPSC_Q_NAME*)
Der Wert ist der Name der Warteschlange für Subskribenten. Er muss mit der Option `QName` übereinstimmen, die in der ursprünglichen Subskription verwendet wurde.

Wenn diese Eigenschaft übergangen wird, ist die Standardeinstellung der Name `ReplyToQ` im Nachrichtendeskriptor (MQMD), der nicht leer sein darf.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Wenn Sie SubName in einem **Deregister Subscriber** -Befehl angeben, hat der Wert für SubName Vorrang vor allen anderen Kennungsfeldern mit Ausnahme der Benutzer-ID, es sei denn, `VariableUserId` ist für die Subskription selbst definiert. Wenn `VariableUserId` nicht festgelegt wird, kann der Befehl **Deregister Subscriber** nur dann erfolgreich ausgeführt werden, wenn die Benutzer-

ID der Befehlsnachricht mit der Benutzer-ID der Subskription übereinstimmt. Andernfalls schlägt der Befehl mit dem Rückkehrcode *MQRCCF_DUPLICATE_IDENTITY* fehl.

Wenn eine Subskription vorhanden ist, die der traditionellen Identität dieses Befehls entspricht, aber keinen SubName hat, schlägt der Befehl **Deregister Subscriber** mit dem Rückkehrcode *MQRCCF_SUB_NAME_ERROR*fehl. Wenn versucht wird, die Registrierung einer Subskription, die den Wert SubName hat, mit einer Befehlsnachricht zurückzunehmen, die mit der traditionellen Identität übereinstimmt, aber den Wert SubName nicht hat, wird der Befehl erfolgreich ausgeführt.

<SubUser> (*MQPSC_SUBSCRIPTION_USER_DATA*)
Dies ist eine Zeichenfolge mit variabler Länge. Der Wert wird vom Warteschlangenmanager zusammen mit der Subskription gespeichert, hat jedoch keinen Einfluss auf die Zustellung von Veröffentlichungen an den Subskribenten. Der Wert kann durch eine erneute Registrierung für dieselbe Subskription mit einem neuen Wert geändert werden. Dieses Attribut ist für die Verwendung durch die Anwendung vorgesehen.

'SubUserData' wird in den Informationen zu den Metathemen (MQCACF_REG_SUB_USER_DATA) für eine Subskription zurückgegeben, falls 'SubUserData' vorhanden ist.

Beispiel

Nachfolgend finden Sie ein Beispiel für den Parameter NameValueData in der Befehlsnachricht für **Deregister Subscriber**. In diesem Beispiel nimmt die Beispielanwendung die Registrierung der Subskription für die Themen zurück, die den letzten Spielstand für alle Spiele enthalten. Als Angaben für die Identität des Subskribenten, einschließlich des Werts CorrelId, werden die Standardeinstellungen aus dem MQMD übernommen.

```
<psc>
 <Command>DeregSub</Command>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Nachricht veröffentlichen

Die Befehlsnachricht **Publish** wird in eine Warteschlange eingereiht oder von einem Warteschlangenmanager an einen Subskribenten übergeben, um Informationen zu einem angegebenen Thema oder zu mehreren angegebenen Themen zu veröffentlichen.

Es werden die Berechtigung zum Einreihen einer Nachricht in eine Warteschlange und die Berechtigung zum Veröffentlichen von Informationen zu einem angegebenen Thema oder zu angegebenen Themen benötigt.

Wenn der Benutzer berechtigt ist, Informationen zu einigen, aber nicht allen Themen zu veröffentlichen, werden nur die betreffenden Themen zur Veröffentlichung verwendet; in einer Warnung werden die Themen genannt, die nicht zur Veröffentlichung verwendet werden.

Liegen für einen Subskribenten übereinstimmende Subskriptionen vor, leitet der Warteschlangenmanager die Nachricht **Publish** an die Warteschlangen für Subskribenten weiter, die in den entsprechenden Befehlsnachrichten für **Register Subscriber** definiert sind.

Weitere Informationen zu den Parametern des Nachrichtendeskriptors (MQMD), die zum Senden einer Befehlsnachricht an den Warteschlangenmanager erforderlich sind und die verwendet werden, wenn ein Warteschlangenmanager eine Veröffentlichung an einen Subskribenten weiterleitet, finden Sie unter Antwortnachricht des Warteschlangenmanagers.

Der Warteschlangenmanager leitet die **Publish** -Nachricht an andere Warteschlangenmanager im Netz weiter, die übereinstimmende Subskriptionen haben, sofern es sich nicht um eine lokale Veröffentlichung handelt.

Eventuell vorhandene Veröffentlichungsdaten werden in den Nachrichtentext übernommen. Die Daten können in einem <mcd>-Ordner im NameValueData-Feld des MQRFH2-Headers beschrieben werden.

Eigenschaften

< Befehl> (*MQPSC_COMMAND*)
Der Wert ist Publish(*MQPSC_PUBLISH*).

Diese Eigenschaft muss angegeben werden.

< Topic> (*MQPSC_TOPIC*)
Bei diesem Wert handelt es sich um eine Zeichenfolge, die das Thema der Veröffentlichung angibt. Es dürfen keine Platzhalterzeichen verwendet werden.

Sie müssen das Thema der Namensliste SYSTEM.QPUBSUB.QUEUE.NAMELIST hinzufügen. Informationen und Anweisungen hierzu finden Sie unter Datenstrom hinzufügen.

Diese Eigenschaft muss angegeben werden; sie kann optional so oft vorkommen, wie Themen vorhanden sind.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
Der Subskriptionspunkt, an dem die Nachricht veröffentlicht werden soll.

In WebSphere Event Broker V6ist der Wert der Eigenschaft <SubPoint> der Wert des Subskriptionspunktattributs des Veröffentlichungsknotens, der die Veröffentlichung verarbeitet.

In WebSphere MQ V7.0.1muss der Wert der Eigenschaft <SubPoint> mit dem Namen eines Subskriptionspunkts übereinstimmen. Siehe Subskriptionspunkt hinzufügen.

<PubOpt> (*MQPSC_PUBLICATION_OPTION*)
Für diese Eigenschaft der Veröffentlichungsoptionen kann einer der folgenden Werte angegeben werden:

RetainPub
(*MQPSC_RETAIN_PUB*)

Der Warteschlangenmanager soll eine Kopie der Veröffentlichung behalten. Ist diese Option nicht gesetzt, wird die Veröffentlichung gelöscht, sobald sie vom Warteschlangenmanager an alle aktuellen Subskribenten gesendet wurde.

IsRetainedPub
(*MQPSC_IS_RETAINED_PUB*)

(Kann nur von einem Warteschlangenmanager festgesetzt werden.) Diese Veröffentlichung wird vom Warteschlangenmanager beibehalten. Der Warteschlangenmanager setzt diese Option, um dem Subskribenten anzuzeigen, dass diese Veröffentlichung bereits gesendet und eine Kopie gespeichert wurde; dies geschieht allerdings nur, wenn die Subskription unter Angabe der Option InformIfRetained angemeldet wurde. Sie wird nur als Antwort auf eine Register Subscriber -oder Request Update -Befehlsnachricht festgelegt. Für ständige Veröffentlichungen, die direkt an Subskribenten gesendet werden, muss diese Option nicht gesetzt werden.

Local
(*MQPSC_LOCAL*)

Diese Option meldet dem Warteschlangenmanager, dass die Veröffentlichung nicht an andere Warteschlangenmanager gesendet werden soll. Diese Veröffentlichung geht an alle in diesem Warteschlangenmanager angemeldeten Subskribenten, sofern eine entsprechende Subskription vorliegt.

OtherSubsOnly
(*MQPSC_OTHER_SUBS_ONLY*)

Diese Option ermöglicht eine einfachere Verarbeitung von Konferenzanwendungen, bei denen eine Veröffentlichungskomponente für ein Thema gleichzeitig auch ein Subskribent für dasselbe Thema ist. Diese Option teilt dem Warteschlangenmanager mit, dass die Veröffentlichung nicht an die Warteschlange für Subskribenten der Veröffentlichungskomponente gesendet werden soll, selbst wenn eine entsprechende Subskription vorliegt. Die Warteschlange für Subskribenten der Veröffentlichungskomponente besteht aus den zugehörigen Optionen QMgrName, QName und optional CorrelId, wie in der folgenden Liste beschrieben.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Die CorrelId im MQMD (die nicht gleich null sein darf) ist in Anwendungen, bei denen der Publisher gleichzeitig auch Subskribent ist, Teil der Warteschlange des Publishers.

None
(*MQPSC_NONE*)

Für alle Optionen werden die Standardwerte verwendet. Dies entspricht dem Übergehen der Eigenschaft für die Veröffentlichungsoptionen. Werden zusätzlich noch andere Optionen angegeben, wird None ignoriert.

Sie können mehr als eine Veröffentlichungsoption verwenden, indem Sie zusätzliche <PubOpt>-Elemente einführen.

Wenn diese Eigenschaft übergangen wird, werden standardmäßig keine Veröffentlichungsoptionen festgelegt.

<PubTime> (*MQPSC_PUBLISH_TIMESTAMP*)
Der Wert ist eine optionale Zeitmarke für die Veröffentlichung, die von der Veröffentlichungskomponente gesetzt wird. Sie umfasst 16 Zeichen und hat folgendes Format:

```
YYYYMMDDHHMMSSTH
```

Die Zeit wird in UT (Universal Time) angegeben. Diese Information wird nicht vom Warteschlangenmanager überprüft, bevor sie an die Subskribenten übermittelt wird.

<SeqNum> (*MQPSC_SEQUENCE_NUMBER*)
Der Wert ist eine optionale Folgenummer, die von der Veröffentlichungskomponente gesetzt wird.

Sie muss bei jeder weiteren Veröffentlichung um 1 erhöht werden. Allerdings wird dies nicht vom Warteschlangenmanager überprüft, er übermittelt diese Information lediglich an die Subskribenten.

Wenn Veröffentlichungen zu ein und demselben Thema in verschiedenen miteinander verbundenen Warteschlangenmanagern veröffentlicht werden, muss die Veröffentlichungskomponente sicherstellen, dass die Folgenummern korrekt sind, sofern sie verwendet werden.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
In Anwendungen, in denen die Veröffentlichungskomponente auch ein Subskribent sein kann (siehe OtherSubsOnly), ist dieser Wert eine Zeichenfolge, die den Namen des Warteschlangenmanagers für die Warteschlange für Subskribenten der Veröffentlichungskomponente angibt.

Wird diese Eigenschaft übergangen, wird standardmäßig der Name ReplyToQMgr im Nachrichtendeskriptor (MQMD) verwendet. Ist dieser Name leer, wird standardmäßig der Name des Warteschlangenmanagers übernommen.

< Warteschlangenname> (*MQPSC_Q_NAME*)
In Anwendungen, in denen der Publisher gleichzeitig auch Subskribent ist (siehe OtherSubsOnly) ist dieser Wert eine Zeichenfolge, die den Namen der Subskribentenwarteschlange des Publishers angibt.

Wird diese Eigenschaft übergangen, wird standardmäßig der Name ReplyToQ im Nachrichtendeskriptor (MQMD) übernommen, der nicht leer sein darf, wenn OtherSubsOnly gesetzt wurde.

Beispiel

Hier einige Beispiele für *NameValueData* für eine **Publish**-Befehlsnachricht.

Beim ersten Beispiel sendet der Spielsimulator in der Beispielanwendung eine Veröffentlichung, die angibt, dass ein Spiel begonnen hat.

```
<psc>
 <Command>Publish</Command>
 <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

Das zweite Beispiel ist ein Beispiel für eine ständige Veröffentlichung. Der aktuelle Stand des Spiels zwischen Team1 und Team2 wird veröffentlicht.

```
<psc>
 <Command>Publish</Command>
 <PubOpt>RetainPub</PubOpt>
 <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Nachricht 'Register Subscriber'

Die Befehlsnachricht für **Register Subscriber** wird von einem Subskribenten oder von einer anderen Anwendung im Auftrag eines Subskribenten an einen Warteschlangenmanager gesendet, um anzuzeigen, dass dieser ein oder mehrere Themen an einem Subskriptionspunkt subskribieren möchte. Außerdem kann ein Filter für Nachrichteninhalte angegeben werden.

In Publish/Subscribe-Filterausdrücken wird die Leistung durch verschachtelte Klammern exponentiell verringert. Vermeiden Sie verschachtelte Klammern mit einer Verschachtelungstiefe größer als 6.

Die Nachricht wird an die Steuerwarteschlange SYSTEM.BROKER.CONTROL.QUEUE des Warteschlangenmanagers gesendet. Neben der Zugriffsberechtigung (festgelegt durch den Systemadministrator des Warteschlangenmanagers) für das Thema bzw. die Themen in der Subskription ist die Berechtigung zum Einreihen einer Nachricht in diese Warteschlange erforderlich.

Wenn der Benutzer die Berechtigung nur für einen Teil der Themen besitzt, werden nur die Themen mit Berechtigung registriert; in einer Warnung werden die nicht registrierten Themen angezeigt.

Im Abschnitt „MQMD-Einstellungen in Befehlsnachrichten für den Warteschlangenmanager” auf Seite 898 finden Sie weitere Informationen zu den Parametern des Nachrichtendeskriptors (MQMD), die zum Senden einer Befehlsnachricht an den Warteschlangenmanager erforderlich sind.

Wenn es sich bei der Warteschlange für Antwortnachrichten um eine temporäre dynamische Warteschlange handelt, wird die Registrierung der Subskription beim Schließen der Warteschlange durch den Warteschlangenmanager automatisch zurückgenommen.

Eigenschaften

< Befehl> (*MQPSC_COMMAND*)
Der Wert ist RegSub (*MQPSC_REGISTER_SUBSCRIBER*). Diese Eigenschaft muss angegeben werden.

< Topic> (*MQPSC_TOPIC*)
Das Thema, zu dem der Subskribent Veröffentlichungen empfangen möchte. Platzhalterzeichen können als Teil des Themas angegeben werden.

Wenn Sie den MQSC-Befehl **display sub** verwenden, um die auf diese Weise erstellte Subskription zu untersuchen, wird der Wert des Tags < Topic> als Eigenschaft TOPICSTR der Subskription angezeigt.

Diese Eigenschaft ist erforderlich und kann optional für die gewünschte Anzahl an Themen wiederholt werden.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
Der Wert ist der Subskriptionspunkt, an den die Subskription angehängt ist.

Wird diese Eigenschaft übergangen, wird der standardmäßige Subskriptionspunkt verwendet.

In WebSphere Event Broker V6muss der Wert der Eigenschaft <SubPoint> mit dem Wert des Subskriptionspunktattributs der subskribierten Veröffentlichungsknoten übereinstimmen.

In WebSphere MQ V7.0.1muss der Wert der Eigenschaft <SubPoint> dem Namen eines Subskriptionspunkts entsprechen. Siehe Subskriptionspunkt hinzufügen.

< Filter > (*MQPSC_FILTER*)
Der Wert ist ein SQL-Ausdruck, der als Filter für die Inhalte von Veröffentlichungsnachrichten verwendet wird. Wenn eine Veröffentlichung im angegebenen Thema mit dem Filter übereinstimmt, wird sie an den Subskribenten gesendet. Diese Eigenschaft entspricht der Auswahlzeichenfolge, die

in MQSUB- und MQOPEN-Aufrufen verwendet wird. Weitere Informationen finden Sie im Abschnitt Auswahl für Inhalt einer Nachricht durchführen.

Wird diese Eigenschaft übergangen, wird der Inhalt nicht gefiltert.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
Für diese Eigenschaft der Registrierungsoptionen kann einer der folgenden Werte angegeben werden:

AddName
(*MQPSC_ADD_NAME*)

Wenn dieser Wert für eine vorhandene Subskription angegeben wird, die mit der traditionellen Identität dieses Befehls 'Register Subscription' übereinstimmt, die jedoch den Wert SubName nicht hat, wird der in diesem Befehl angegebene Wert SubName der Subskription hinzugefügt.

Wenn AddName angegeben ist, ist das Feld SubName obligatorisch; andernfalls wird MQRCCF_REG_OPTIONS_ERROR zurückgegeben.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Die CorrelId im Nachrichtendeskriptor (MQMD) wird beim Senden von übereinstimmenden Veröffentlichungen an die Warteschlange für Subskribenten verwendet. Die CorrelId darf nicht null sein.

FullResp
(*MQPSC_FULL_RESPONSE*)

Wenn dieser Wert angegeben ist, werden in der Antwortnachricht alle Attribute der Subskription zurückgegeben, sofern der Befehl nicht fehlschlägt.

Der Wert 'FullResp' ist nur gültig, wenn sich die Befehlsnachricht auf eine einzelne Subskription bezieht. Deshalb ist nur ein Thema im Befehl zulässig; andernfalls schlägt der Befehl mit dem Rückkehrcode *MQRCCF_REG_OPTIONS_ERROR* fehl.

InformIfRet
(*MQPSC_INFORM_IF_RETAINED*)

Der Warteschlangenmanager informiert den Subskribenten darüber, ob eine Veröffentlichung beibehalten wird, wenn er eine Publish-Nachricht als Antwort auf eine **Register Subscriber** -oder **Request Update** -Befehlsnachricht sendet. Der Warteschlangenmanager integriert dazu die Veröffentlichungsoption IsRetainedPub in die Nachricht.

JoinExcl
(*MQPSC_JOIN_EXCLUSIVE*)

Mit dieser Option wird angezeigt, dass der angegebene Wert SubIdentity als exklusives Element des Identitätssatzes für die Subskription hinzugefügt werden soll und dass dem Satz keine weiteren Identitäten hinzugefügt werden können.

Wenn die Identität bereits für die gemeinsame Nutzung verknüpft wurde und der einzige Eintrag im Satz ist, wird der Satz exklusiv für diese Identität gesperrt. Wenn sich jedoch weitere Identitäten im Identitätssatz (mit gemeinsamem Zugriff) für die Subskription befinden, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_SUBSCRIPTION_IN_USE* fehl.

JoinShared
(*MQPSC_JOIN_SHARED*)

Mit dieser Option wird angezeigt, dass der angegebene Wert SubIdentity dem Identitätssatz für die Subskription hinzugefügt werden soll.

Wenn für die Subskription aktuell eine exklusive Sperre festgelegt wurde (mithilfe der Option JoinExcl), schlägt der Befehl mit dem Rückkehrcode *MQRCCF_SUBSCRIPTION_LOCKED* fehl, es sei denn, dass die zur Sperre gehörige Identität mit der in der Befehlsnachricht genannten Identität übereinstimmt. In diesem Fall wird die exklusive Sperre automatisch in eine gemeinsame Sperre geändert.

Local
(*MQPSC_LOCAL*)

Die Subskription ist lokal und wird nicht an andere Warteschlangenmanager im Netz verteilt. Die in anderen Warteschlangenmanagern vorgenommenen Veröffentlichungen werden diesem Subskribenten nicht zugestellt, sofern er nicht auch über eine entsprechende globale Subskription verfügt.

NewPubsOnly
(*MQPSC_NEW_PUBS_ONLY*)

Ständige Veröffentlichungen, die bei der Registrierung der Subskription bereits vorhanden sind, werden nicht an den Subskribenten gesendet. Es werden nur neue Veröffentlichungen gesendet.

Wenn ein Subskribent sich erneut registriert und diese Option aufhebt, erhält er möglicherweise eine Veröffentlichung erneut, die bereits vorher an ihn gesendet wurde.

NoAlter
(*MQPSC_NO_ALTER*)

Die Attribute einer vorhandenen übereinstimmenden Subskription werden nicht geändert.

Diese Option wird bei der Erstellung einer Subskription ignoriert. Alle anderen angegebenen Optionen gelten für die neue Subskription.

Wenn für einen Wert `SubIdentity` auch eine der Join-Optionen (`JoinExcl` oder `JoinShared`) angegeben ist, wird die Identität dem Identitätssatz hinzugefügt, unabhängig davon, ob `NoAlter` angegeben ist.

None
(*MQPSC_NONE*)

Für alle Optionen zur Registrierung werden die Standardwerte angegeben.

Wenn der Subskribent bereits registriert ist, werden seine Optionen auf ihre Standardwerte zurückgesetzt (beachten Sie, dass dies *nicht* dieselbe Auswirkung hat wie das Auslassen der Eigenschaft für Registrierungsoptionen), und der Subskriptionsablauf wird vom MQMD der **`Register Subscriber`** -Nachricht aktualisiert.

Wenn gleichzeitig andere Registrierungsoptionen angegeben werden, wird die Option None ignoriert.

NonPers
(*MQPSC_NON_PERSISTENT*)

Veröffentlichungen, die mit dieser Subskription übereinstimmen, werden dem Subskribenten als nicht persistente Nachrichten zugestellt.

Pers
(*MQPSC_PERSISTENT*)

Veröffentlichungen, die mit dieser Subskription übereinstimmen, werden dem Subskribenten als persistente Nachrichten zugestellt.

PersAsPub
(*MQPSC_PERSISTENT_AS_PUBLISH*)

Veröffentlichungen, die mit dieser Subskription übereinstimmen, werden dem Subskribenten mit der von der Veröffentlichungskomponente angegebenen Persistenz zugestellt. Dies ist das Standardverhalten.

PersAsQueue
(*MQPSC_PERSISTENT_AS_Q*)

Veröffentlichungen, die mit dieser Subskription übereinstimmen, werden dem Subskribenten mit der in der Warteschlange für Subskribenten angegebenen Persistenz zugestellt.

PubOnReqOnly
(*MQPSC_PUB_ON_REQUEST_ONLY*)

Der Warteschlangenmanager sendet keine Veröffentlichungen an den Subskribenten, außer als Antwort auf eine Befehlsnachricht für **Request Update**.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Wenn dieser Wert angegeben ist, wird die Identität des Subskribenten (Warteschlange, Warteschlangenmanager und Korrelations-ID) nicht auf eine einzige Benutzer-ID eingeschränkt. Dies unterscheidet sich vom bestehenden Verhalten des Warteschlangenmanagers, der die Benutzer-ID der ursprünglichen Registrierungsnachricht der Identität des Subskribenten zuordnet und anschließend verhindert, dass andere Benutzer diese Identität verwenden. Wenn ein neuer Subskribent die gleiche Identität verwenden möchte, wird der Rückkehrcode *MQRCCF_DUPLICATE_SUBSCRIPTION* zurückgegeben.

Dadurch können alle Benutzer die Subskription ändern oder die Registrierung zurücknehmen, wenn sie über die entsprechende Berechtigung verfügen. Deshalb muss nicht überprüft werden, ob die Benutzer-ID mit der des ursprünglichen Subskribenten übereinstimmt.

Um diese Option einer vorhandenen Subskription hinzuzufügen, muss der Befehl mit der gleichen Benutzer-ID ausgegeben werden wie die ursprüngliche Subskription.

Wenn für die Subskription des Befehls **Request Update** die Option `VariableUserId` festgelegt ist, muss diese mit der Uhrzeit der Aktualisierungsanforderung festgelegt sein, um anzuzeigen, auf welche Subskription sie sich bezieht. Andernfalls wird die Benutzer-ID des Befehls **Request Update** zur Ermittlung der Subskription verwendet. Diese ID wird zusammen mit den anderen Subskribenten-IDs überschrieben, falls ein Subskriptionsname bereitgestellt wird.

Wenn sich eine Befehlsnachricht für **Register Subscriber**, in der diese Option nicht angegeben ist, auf eine vorhandene Subskription bezieht, für die diese Option festgelegt ist, wird die Option aus dieser Subskription entfernt und die Benutzer-ID der Subskription ist dann fest zugeordnet. Wenn bereits ein Subskribent mit der gleichen Identität (Warteschlange, Warteschlangenmanager und Korrelations-ID) vorhanden ist, dem aber eine andere Benutzer-ID zugeordnet ist, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_DUPLICATE_IDENTITY* fehl, da einer Subskribentenidentität nur eine Benutzer-ID zugeordnet sein kann.

Wenn die Eigenschaft für die Registrierungsoptionen übergangen wurde und der Subskribent bereits registriert ist, werden die zugehörigen Registrierungsoptionen nicht geändert und das Ablaufdatum der Subskription wird aus dem Nachrichtendeskriptor MQMD der Nachricht **Register Subscriber** aktualisiert.

Wenn der Subskribent nicht bereits registriert ist, wird eine neue Subskription erstellt, in der für alle Registrierungsoptionen Standardwerte verwendet werden.

Die Standardwerte sind `PersAsPub` und es sind keine weiteren Optionen festgelegt.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
Der Wert ist der Name des Warteschlangenmanagers für die Warteschlange für Subskribenten, an die der Warteschlangenmanager übereinstimmende Veröffentlichungen sendet.

Wird diese Eigenschaft übergangen, wird standardmäßig der Name `ReplyToQMgr` im Nachrichtendeskriptor (MQMD) verwendet. Wenn der sich daraus ergebende Name leer ist, wird standardmäßig der Wert `QMgrName` des Warteschlangenmanagers verwendet.

< Warteschlangenname> (*MQPSC_Q_NAME*)
Der Wert ist der Name der Warteschlange für Subskribenten, an die der Warteschlangenmanager übereinstimmende Veröffentlichungen sendet.

Wird diese Eigenschaft übergangen, ist die Standardeinstellung der Name `ReplyToQ` im Nachrichtendeskriptor (MQMD), der in diesem Fall nicht leer sein darf.

Wenn die Warteschlange eine temporäre dynamische Warteschlange ist, muss die nicht persistente Zustellung von Veröffentlichungen (`NonPers`) in der Eigenschaft <RegOpt> angegeben werden.

Wenn es sich bei der Warteschlange um eine temporäre dynamische Warteschlange handelt, wird die Registrierung der Subskription beim Schließen der Warteschlange durch den Warteschlangenmanager automatisch zurückgenommen.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Dies ein Name, der einer bestimmten Subskription zugeordnet wird. Sie können diesen Namen anstelle des Warteschlangenmanagers, der Warteschlange und der optionalen Korrelations-ID verwenden, um sich auf eine Subskription zu beziehen.

Wenn bereits eine Subskription mit diesem **SubName**vorhanden ist, werden alle anderen Attribute der Subskription (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData und Expiry) mit den Attributen (sofern angegeben) überschrieben, die in der neuen `Register Subscriber` -Befehlsnachricht übergeben werden. Wenn für **SubName** jedoch kein Feld 'QName' angegeben ist und 'ReplyToQ' im MQMD-Header angegeben ist, wird die Warteschlange für Subskribenten in 'ReplyToQ' geändert.

Wenn bereits eine Subskription vorhanden ist, die mit der traditionellen Identität dieses Befehls übereinstimmt, für die jedoch keine Eigenschaft **SubName** angegeben ist, schlägt der Befehl zur Registrierung mit dem Rückkehrcode *MQRCCF_DUPLICATE_SUBSCRIPTION* fehl, sofern die Option **AddName** nicht angegeben ist.

Wenn Sie versuchen, eine vorhandene benannte Subskription zu ändern, indem Sie einen anderen `Register Subscriber` -Befehl verwenden, der denselben **SubName**angibt, und die Werte von Topic, QMgrName, QName und CorrelId im neuen Befehl mit einer anderen vorhandenen Subskription übereinstimmen, mit oder ohne definierten SubName , schlägt der Befehl mit dem Rückkehrcode *MQRCCF_DUPLICATE_SUBSCRIPTION*fehl. Dadurch wird verhindert, dass sich die Namen von zwei Subskriptionen auf die gleiche Subskription beziehen.

<SubIdentity> (*MQPSC_SUBSCRIPTION_IDENTITY*)
Diese Zeichenfolge dient zur Darstellung einer Anwendung, die Interesse an einer Subskription hat. Es handelt sich dabei um eine optionale Zeichenfolge variabler Länge mit einer maximalen Länge von 64 Zeichen. Der Warteschlangenmanager verwaltet für jede Subskription einen Satz an Subskribentenidentitäten. Jede Subskription kann entweder nur eine Identität oder eine unbegrenzte Anzahl von Identitäten enthalten (weitere Informationen finden Sie in den Optionen **JoinShared** und **JoinExcl**).

Mit einem Subskriptionsbefehl, in dem die Option **JoinShared** oder **JoinExcl** angegeben ist, wird die Eigenschaft **SubIdentity** dem Identitätssatz der Subskription hinzugefügt, sofern sie nicht bereits vorhanden ist und die vorhandenen Identitätssätze diese Aktion zulassen. Es darf also kein anderer Benutzer exklusiv zugeordnet worden sein und der Identitätssatz darf nicht leer sein.

Jede Änderung der Attribute der Subskription als Ergebnis eines Befehls `Register Subscriber` , in dem eine **SubIdentity** angegeben ist, ist nur erfolgreich, wenn sie das einzige Mitglied der Gruppe von Identitäten für diese Subskription ist. Andernfalls schlägt der Befehl mit dem Rückkehrcode *MQRCCF_SUBSCRIPTION_IN_USE* fehl. Dadurch können die Subskriptionsattribute nicht geändert werden, ohne dass andere interessierte Subskribenten informiert werden.

Wenn Sie eine Zeichenfolge mit einer Länge von mehr als 64 Zeichen angeben, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_SUB_IDENTITY_ERROR* fehl.

<SubUser> (*MQPSC_SUBSCRIPTION_USER_DATA*)
Dies ist eine Zeichenfolge mit variabler Länge. Der Wert wird vom Warteschlangenmanager zusammen mit der Subskription gespeichert, hat jedoch keinen Einfluss auf die Zustellung von Veröffentlichungen an den Subskribenten. Der Wert kann durch eine erneute Registrierung für dieselbe Subskription mit einem neuen Wert geändert werden. Dieses Attribut ist für die Verwendung durch die Anwendung vorgesehen.

SubUserData wird in den Informationen zu den Metathemen (*MQCACF_REG_SUB_USER_DATA*) für eine Subskription zurückgegeben, falls 'SubUserData' vorhanden ist.

Wenn Sie mehrere Registrierungsoptionswerte `NonPers, PersAsPub, PersAsQueue, and Persan`-geben, wird nur der letzte Wert verwendet. Diese Optionen können nicht in einer einzelnen Subskription kombiniert werden.

Beispiel

Nachfolgend finden Sie ein Beispiel für den Parameter NameValueData in der Befehlsnachricht für **Register Subscriber**. In der Beispielanwendung registriert der Ergebnisdienst mit dieser Befehlsnachricht eine Subskription zu den Themen, die die letzten Spielstände aller Spiele enthalten. Dabei wird die Option 'Persistent as publish' angegeben. Als Angaben für die Identität des Subskribenten, einschließlich des Werts CorrelId, werden die Standardeinstellungen aus dem MQMD übernommen.

```
<psc>
 <Command>RegSub</Command>
 <RegOpt>PersAsPub</RegOpt>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Nachricht 'Request Update'

Die Befehlsnachricht für **Request Update** wird von einem Subskribenten an einen Warteschlangenmanager gesendet, um die aktuellen ständigen Veröffentlichungen für das angegebene Thema und den Subskriptionspunkt anzufordern, die mit dem gegebenen (optionalen) Filter übereinstimmen.

Diese Nachricht wird an die Steuerwarteschlange *SYSTEM.BROKER.CONTROL.QUEUE* des Warteschlangenmanagers gesendet. Die Berechtigung zum Einreihen einer Nachricht in diese Warteschlange ist erforderlich. Darüber hinaus legt der Systemadministrator des Warteschlangenmanagers die Zugriffsberechtigung für das Thema in der Aktualisierungsanforderung fest.

Dieser Befehl wird normalerweise verwendet, wenn der Subskribent die Option PubOnReqOnly bei der Registrierung angegeben hat. Wenn der Warteschlangenmanager über übereinstimmende ständige Veröffentlichungen verfügt, werden diese an den Subskribenten gesendet. Wenn der Warteschlangenmanager keine übereinstimmenden ständigen Veröffentlichungen enthält, schlägt die Anforderung mit dem Rückkehrcode *MQRCCF_NO_RETAINED_MSG* fehl. Die anfordernde Stelle muss zuvor eine Subskription mit den gleichen Werten für 'Topic', 'SubPoint' und 'Filter' registriert haben.

Eigenschaften

< Befehl> (*MQPSC_COMMAND*)
Der Wert ist ReqUpdate (*MQPSC_REQUEST_UPDATE*). Diese Eigenschaft muss angegeben werden.

< Topic> (*MQPSC_TOPIC*)
Der Wert ist das Thema, das der Subskribent anfordert; Platzhalterzeichen sind zulässig.

Diese Eigenschaft muss angegeben werden, es ist allerdings nur ein Auftreten in dieser Nachricht zulässig.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
Der Wert ist der Subskriptionspunkt, an den die Subskription angehängt ist.

Wird diese Eigenschaft übergangen, wird der standardmäßige Subskriptionspunkt verwendet.

< Filter > (*MQPSC_FILTER*)
Der Wert ist ein ESQL-Ausdruck, der als Filter für die Inhalte von Veröffentlichungsnachrichten verwendet wird. Wenn eine Veröffentlichung im angegebenen Thema mit dem Filter übereinstimmt, wird sie an den Subskribenten gesendet.

Die Eigenschaft <Filter> sollte denselben Wert haben wie der, der in der ursprünglichen Subskription angegeben wurde, für die Sie jetzt ein Update anfordern.

Wird diese Eigenschaft übergangen, wird der Inhalt nicht gefiltert.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
Die Eigenschaften für die Registrierungsoptionen können folgenden Wert haben:

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Die CorrelId im Nachrichtendeskriptor (MQMD), die nicht den Wert null haben darf, wird beim Senden von übereinstimmenden Veröffentlichungen an die Warteschlange für Subskribenten verwendet.

None
(*MQPSC_NONE*)

Für alle Optionen werden die Standardwerte verwendet. Dies hat den gleichen Effekt wie das Weglassen der Eigenschaft <RegOpt>. Wenn gleichzeitig andere Optionen angegeben werden, wird die Option None ignoriert.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Wenn dieser Wert angegeben ist, wird die Identität des Subskribenten (Warteschlange, Warteschlangenmanager und Korrelations-ID) nicht auf eine einzige Benutzer-ID eingeschränkt. Dies unterscheidet sich vom bestehenden Verhalten des Warteschlangenmanagers, der die Benutzer-ID der ursprünglichen Registrierungsnachricht der Identität des Subskribenten zuordnet und anschließend verhindert, dass andere Benutzer diese Identität verwenden. Wenn ein neuer Subskribent die gleiche Identität verwenden möchte, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_DUPLICATE_SUBSCRIPTION* fehl.

Dadurch können alle Benutzer die Subskription ändern oder die Registrierung zurücknehmen, wenn sie über die entsprechende Berechtigung verfügen. Deshalb muss nicht überprüft werden, ob die Benutzer-ID mit der des ursprünglichen Subskribenten übereinstimmt.

Um diese Option einer vorhandenen Subskription hinzuzufügen, muss der Befehl mit der gleichen Benutzer-ID ausgegeben werden wie die ursprüngliche Subskription.

Wenn für die Subskription des Befehls **Request Update** die Option VariableUserId festgelegt ist, muss diese mit der Uhrzeit der Aktualisierungsanforderung festgelegt sein, um anzuzeigen, auf welche Subskription sie sich bezieht. Andernfalls wird die Benutzer-ID des Befehls **Request Update** zur Ermittlung der Subskription verwendet. Diese ID wird zusammen mit den anderen Subskribenten-IDs überschrieben, falls ein Subskriptionsname bereitgestellt wird.

Wenn diese Eigenschaft übergangen wird, werden standardmäßig keine Registrierungsoptionen festgelegt.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
Der Wert ist der Name des Warteschlangenmanagers für die Warteschlange für Subskribenten, an die der Warteschlangenmanager die übereinstimmende ständige Veröffentlichung sendet.

Wird diese Eigenschaft übergangen, wird standardmäßig der Name ReplyToQMgr im Nachrichtendeskriptor (MQMD) verwendet. Wenn der sich daraus ergebende Name leer ist, wird standardmäßig der Wert QMgrName des Warteschlangenmanagers verwendet.

< Warteschlangenname> (*MQPSC_Q_NAME*)
Der Wert ist der Name der Warteschlange für Subskribenten, an die der Warteschlangenmanager die übereinstimmende ständige Veröffentlichung sendet.

Wird diese Eigenschaft übergangen, ist die Standardeinstellung der Name ReplyToQ im Nachrichtendeskriptor (MQMD), der in diesem Fall nicht leer sein darf.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Dies ein Name, der einer bestimmten Subskription zugeordnet wird. Wenn der Name im Befehl **Request Update** angegeben wurde, hat der Wert SubName Vorrang vor allen anderen ID-Feldern außer der Benutzer-ID, es sei denn, dass der Wert VariableUserId in der Subskription selbst festgelegt wird. Wenn VariableUserId nicht festgelegt wird, kann der Befehl *Request Update* nur dann erfolgreich ausgeführt werden, wenn die Benutzer-ID der Befehlsnachricht mit der Benutzer-ID der Subskription übereinstimmt. Wenn die Benutzer-ID der Befehlsnachricht nicht mit der Benutzer-ID der Subskription übereinstimmt, schlägt der Befehl mit dem Rückkehrcode *MQRCCF_DUPLICATE_IDENTITY* fehl.

Wenn die Option VariableUserId festgelegt ist und die Benutzer-ID von der Benutzer-ID der Subskription abweicht, wird der Befehl nur dann erfolgreich ausgeführt, wenn die Benutzer-ID der

neuen Befehlsnachricht über die Berechtigung zum Durchsuchen der Datenstromwarteschlange und zum Einreihen von Nachrichten in die Warteschlange für Subskribenten für diese Subskription verfügt. Andernfalls schlägt der Befehl mit dem Rückkehrcode *MQRCCF_NOT_AUTHORIZED* fehl.

Wenn eine Subskription vorhanden ist, die der traditionellen Identität dieses Befehls entspricht, aber keinen SubNamehat, schlägt der**Request Update** -Befehl mit dem Rückkehrcode *MQRCCF_SUB_NAME_ERROR*fehl.

Wenn versucht wird, die Aktualisierung einer Subskription, die den Wert SubName hat, mit einer Befehlsnachricht anzufordern, die mit der traditionellen Identität übereinstimmt, aber den Wert SubName nicht hat, wird der Befehl erfolgreich ausgeführt.

Beispiel

Nachfolgend ein Beispiel für NameValueData für die Befehlsnachricht für **Request Update**. In der Beispielanwendung fordert der Ergebnisdienst mit dieser Befehlsnachricht eine ständige Veröffentlichung an, die die letzten Spielstände aller Teams enthält. Als Angaben für die Identität des Subskribenten, einschließlich des Werts CorrelId, werden die Standardeinstellungen aus dem MQMD übernommen.

```
<psc>
 <Command>ReqUpdate</Command>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Nachricht 'Queue Manager Response'

Die Nachricht **Queue Manager Response** wird von einem Warteschlangenmanager an die Warteschlange ReplyToQ einer Veröffentlichungskomponente oder eines Subskribenten gesendet, um anzuzeigen, ob eine vom Warteschlangenmanager empfangene Befehlsnachricht erfolgreich ausgeführt wurde oder fehlgeschlagen ist. Diese Aktion wird ausgeführt, wenn durch den Deskriptor für Befehlsnachrichten angegeben wurde, dass eine Antwort erforderlich ist.

Die Antwortnachricht befindet sich im NameValueData-Feld des MQRFH2-Headers, in einem <pscr>-Ordner.

Im Falle einer Warnung oder eines Fehlers enthält die Antwortnachricht den Ordner <psc> aus der Befehlsnachricht sowie den Ordner <pscr>. Falls Nachrichtendaten vorhanden sind, befinden sich diese nicht in der Antwortnachricht des Warteschlangenmanagers. Im Falle eines Fehlers wird keine der Nachrichten verarbeitet, durch die ein Fehler verursacht wurde; im Falle einer Warnung werden einige der Nachrichten möglicherweise erfolgreich verarbeitet.

Falls durch einen Fehler eine Antwort gesendet wird, geschieht Folgendes:

- Bei Veröffentlichungsnachrichten versucht der Warteschlangenmanager, die Antwort an die Warteschlange für nicht zustellbare Nachrichten von WebSphere MQ zu senden, falls der MQPUT-Befehl fehlschlägt. Dadurch kann die Veröffentlichung selbst dann an Subskribenten gesendet werden, wenn die Antwort nicht an die Veröffentlichungskomponente zurückgesendet werden kann.
- Bei anderen Nachrichten oder falls die Veröffentlichungsantwort nicht an die Warteschlange für nicht zustellbare Nachrichten gesendet werden kann, wird ein Fehler protokolliert und die Befehlsnachricht wird normalerweise rückgängig gemacht. Diese Aktion ist davon abhängig, wie der MQInput-Knoten konfiguriert wurde.

Eigenschaften

< Completion> (*MQPSCR_COMPLETION*)
Der Beendigungscode, der einen der folgenden drei Werte annehmen kann:

ok
Befehl erfolgreich beendet.

warning
Befehl beendet, aber mit Warnung.

error
Befehl fehlgeschlagen.

< Antwort> (*MQPSCR_RESPONSE*)
Dies ist die Antwort auf eine Befehlsnachricht, falls dieser Befehl den Beendigungscode `warning` (Warnung) oder `error` (Fehler) erzeugt hat. Sie enthält eine <Reason>-Eigenschaft und kann andere Eigenschaften enthalten, die die Ursache für die Warnung oder den Fehler anzeigen.

Im Falle eines oder mehrerer Fehler wird nur ein Antwortordner erstellt, in dem nur die Ursache des ersten Fehlers angezeigt wird. Im Falle einer oder mehrerer Warnungen wird ein Antwortordner für jede Warnung erstellt.

< Ursache> (*MQPSCR_REASON*)
Der Ursachencode, durch den der Beendigungscode näher bestimmt wird, falls es sich um den Beendigungscode `warning` (Warnung) oder `error` (Fehler) handelt. Es wird einer der im folgenden Beispiel aufgeführten Fehlercodes festgelegt. Die Eigenschaft <Reason> ist in einem <Response>-Ordner enthalten. Auf den Ursachencode kann eine beliebige gültige Eigenschaft aus dem Ordner <psc> (z. B. ein Themenname) folgen, die die Ursache für den Fehler oder die Warnung angibt. Wenn Sie einen Ursachencode von? ???, Überprüfen Sie die Daten auf Richtigkeit, z. B. übereinstimmende spitze Klammern (< >).

Beispiele

Nachfolgend finden Sie einige Beispiele von `NameValueData` in einer **Queue Manager Response**-Nachricht. Beispiel für eine Erfolgsantwort:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Nachfolgend finden Sie ein Beispiel einer Fehlerantwort; bei dem Fehler handelt es sich um einen Filterfehler. Die erste Zeichenfolge `NameValueData` enthält die Antwort, die zweite Zeichenfolge enthält den ursprünglichen Befehl.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Nachfolgend finden Sie ein Beispiel einer Warnungsantwort (aufgrund unberechtigter Themen). Die erste Zeichenfolge `NameValueData` enthält die Antwort, die zweite Zeichenfolge `NameValueData` enthält den ursprünglichen Befehl.

```
<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>
```

Publish/Subscribe-Ursachencodes

Diese Ursachencodes können im Feld `Ursache` eines Veröffentlichung/Abonnement-Antwortordners <pscr> zurückgegeben werden. Es werden auch Konstanten aufgeführt, mit denen diese Codes in den Programmiersprachen C oder C++ dargestellt werden.

Für 'MQRC_'-Konstanten ist die WebSphere MQ-Headerdatei `cmqc.h` erforderlich. Für die 'MQRCCF_'-Konstanten ist die WebSphere MQ-Headerdatei `cmqcfc.h` erforderlich (neben *MQRCCF_FILTER_ERROR* und *MQRCCF_WRONG_USER*, für die die Headerdatei `cmqpsc.h` erforderlich ist).

Ursachencode und Text	Beschreibung	Ausgegeben von
2336 MQRC_RFH_COMMAND_ERROR	Gültige Werte für das Feld < Command> eines <psc> -Ordners sind RegSub, DeregSub, Publish, DeletePubund ReqUpdate. Bei allen anderen Werten wird dieser Fehlercode ausgegeben.	Jeder Befehl
2337 MQRC_RFH_ PARM_ERROR	Die Ordner <psc> und <mcd> verfügen beide über eine Gruppe gültiger Parameter, die in ihnen angegeben werden können. Prüfen Sie die Beschreibungen zu diesen Ordnern und stellen Sie sicher, dass Sie keine falschen Parameter angegeben haben.	Jeder Befehl
2338 MQRC_RFH_DUPLICATE_PARM	Einige Parameter (z. B. Thema) innerhalb eines <psc>-Ordners können wiederholt werden, andere (z. B. Befehl) können jedoch nicht wiederholt werden. Stellen Sie sicher, dass keine nicht wiederholbaren Parameter doppelt vorhanden sind.	Jeder Befehl
2339 MQRC_RFH_PARM_MISSING	Einige Parameter innerhalb von <psc> oder <mcd>-Ordnern sind optional und können weggelassen werden; einige sind obligatorisch und dürfen nicht weggelassen werden. Stellen Sie sicher, dass Sie alle obligatorischen Parameter in Ihren <psc> und <mcd>-Ordnern eingeschlossen haben.	Jeder Befehl
2551 MQRC_SELECTION_NOT_AVAILABLE	Es war kein Provider für erweiterte Nachrichtenauswahl verfügbar, der ermitteln kann, welche Subskribenten mit einem Filter die Veröffentlichung empfangen sollen.	Befehle 'Publizieren', 'Subskribent registrieren' und 'Request Update'
	Es war kein Provider für erweiterte Nachrichtenauswahl zur Verarbeitung des Filters für den angegebenen Subskribenten verfügbar.	Befehle 'Subskribent registrieren' und 'Request Update'

Ursachencode und Text	Beschreibung	Ausgegeben von
2554 MQRC_CONTENT_ERROR	Ein Provider für erweiterte Nachrichtenauswahl hat einen Fehler in der aktuellen oder ständigen Veröffentlichung ermittelt.	Befehle 'Publizieren' und 'Request Update'
3008 MQRCCF_COMMAND_FAILED	Es ist ein interner Fehler aufgetreten, durch den der Befehl nicht ordnungsgemäß ausgeführt werden konnte. Der Fehler kann auftreten, wenn der Befehl erneut ausgegeben wird. Das Systemereignisprotokoll für den Warteschlangenmanager enthält Informationen, die Sie beim Melden des Problems bei IBM verwenden sollten.	Jeder Befehl
3072 MQRCCF_TOPIC_ERROR	Mindestens einer der Werte, den Sie für den Parameter 'Topic' bereitgestellt haben, ist falsch. Stellen Sie sicher, dass Ihre Werte für 'Topic' den angegebenen Einschränkungen entsprechen.	Jeder Befehl
3073 MQRCCF_NOT_REGISTERED	Die Kombination aus 'SubPoint', 'Topic' und 'Filter', die Sie im Befehl 'DeregSub' oder 'ReqUpdate' angegeben haben, war entweder keine Kombination, mit der Sie sich zuvor registriert hatten, oder, falls die Option 'DeregAll' für den Befehl 'DeregSub' angegeben wurde, eine der Eigenschaften 'SubPoint', 'Topic' oder 'Filter' wurde nicht zur Zurücknahme der Registrierung einer Subskription verwendet.	Befehle 'Registrierung von Subskribent zurücknehmen' und 'Request Update'
3074 MQRCCF_Q_MGR_NAME_ERROR	Der angegebene Warteschlangenmanager war nicht gültig bzw. der Warteschlangenmanager war nicht verfügbar oder nicht vorhanden.	Befehle 'Registrierung von Subskribent zurücknehmen', 'Publizieren', 'Subskribent registrieren' und 'Request Update'
3076 MQRCCF_Q_NAME_ERROR	Der angegebene Warteschlangenname war nicht gültig oder die Warteschlange war nicht im angegebenen Warteschlangenmanager vorhanden.	Befehle 'Registrierung von Subskribent zurücknehmen', 'Publizieren', 'Subskribent registrieren' und 'Request Update'
3077 MQRCCF_NO_RETAINED_MSG	Es gab keine Nachrichten für ständige Veröffentlichungen für das angegebene Thema. Je nach dem Aufbau Ihres Anwendungsprogramms kann es sich dabei um einen Fehler handeln.	Befehl 'Request Update'

Ursachencode und Text	Beschreibung	Ausgegeben von
3079 MQRCCF_INCORRECT_Q	Die Befehle 'RegSub', 'DeregSub' und 'ReqUpdate' werden immer an die Warteschlange SYSTEM.BROKER.CONTROL.QUEUE des Warteschlangenmanagers gesendet, für den sie gedacht waren. Die Befehle 'Publizieren' und 'Delete Publication' werden an die Eingabewarteschlange des jeweiligen Publish/Subscribe-Nachrichtenflusses gesendet, für den sie gedacht waren; dies wird bei der Entwicklung des Nachrichtenflusses festgelegt. Dieser Fehlercode wird zurückgegeben, wenn ein Befehl an die falsche Warteschlange gesendet wird.	Jeder Befehl
3080 MQRCCF_CORREL_ID_ERROR	Sie haben 'CorrelAsId' als einen Ihrer 'RegOpt'-Parameter angegeben. Das Feld 'CorrelId' von MQMD enthält aber keine gültige Korrelations-ID (d. h., es wird auf MQCI_NONE gesetzt).	Befehle 'Registrierung von Subskribent zurücknehmen' und 'Subskribent registrieren'
3081 MQRCCF_NOT_AUTHORIZED	Sie haben keine Berechtigung zum Ausführen der angeforderten Aktion. Die Berechtigungseinstellungen für den Warteschlangenmanager werden vom Systemadministrator mithilfe des Editors für die Themenhierarchie vorgenommen.	Befehle 'Publizieren' und 'Subskribent registrieren'
3083 MQRCCF_REG_OPTIONS_ERROR	Im Ordner <psc>, der den Befehl RegSub oder DeregSub enthält, wurde ein nicht erkannter Parameter „RegOpt“ angegeben.	Befehle 'Registrierung von Subskribent zurücknehmen' und 'Subskribent registrieren'
3084 MQRCCF_PUB_OPTIONS_ERROR	Sie haben einen nicht erkannten Parameter „PubOpt“ im Ordner <psc> angegeben, der den Befehl „Veröffentlichen“ enthält.	Befehl 'Publizieren'
3087 MQRCCF_DEL_OPTIONS_ERROR	Sie haben einen nicht erkannten Parameter DelOpt im Ordner <psc> angegeben, der den Befehl DeletePub enthält.	Befehl 'Delete Publication'
3150 MQRCCF_FILTER_ERROR	Der für den Filterparameter angegebene Wert ist nicht gültig. Prüfen Sie den Abschnitt, in dem die gültige Syntax für Filterausdrücke beschrieben wird, und stellen Sie sicher, dass Ihr Ausdruck den Anforderungen entspricht.	Befehle 'Registrierung von Subskribent zurücknehmen', 'Subskribent registrieren' und 'Request Update'

Ursachencode und Text	Beschreibung	Ausgegeben von
3151 MQRCCF_WRONG_USER	Es ist bereits eine Subskription vorhanden, die mit der angegebenen Subskription übereinstimmt; diese wurde jedoch von einem anderen Benutzer registriert. Eine Änderung oder Zurücknahme der Registrierung für eine Subskription kann nur von dem Benutzer vorgenommen werden, der die Subskription ursprünglich registrierte.	Befehle 'Registrierung von Subskribent zurücknehmen', 'Subskribent registrieren' und 'Request Update'
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Es ist bereits eine übereinstimmende Subskription mit einem anderen Subskriptionsnamen vorhanden.	
3153 MQRCCF_SUB_NAME_ERROR	Entweder ist das Format eines Subskriptionsnamens nicht gültig oder es ist bereits eine übereinstimmende Subskription ohne Subskriptionsname vorhanden.	
3154 MQRCCF_SUB_IDENTITY_ERROR	Der Parameter für die Identität der Subskription ist fehlerhaft. Entweder überschreitet der angegebene Wert die maximal zulässige Länge oder die Identität der Subskription ist derzeit kein Mitglied des Identitätssatzes für die Subskription und es wurde keine Join-Registrierungsoption angegeben.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Ein Mitglied des Identitätssatzes, bei dem es sich nicht um das einzige Mitglied dieses Satzes handelt, wollte eine Subskription ändern oder die Registrierung einer Subskription zurücknehmen.	
3156 MQRCCF_SUBSCRIPTION_LOCKED	Die Subskription wird durch eine andere Identität exklusiv gesperrt.	
3157 MQRCCF_ALREADY_JOINED	Es wurde eine Join-Registrierungsoption angegeben, aber die Subskribentenidentität war bereits Mitglied des Identitätssatzes für die Subskription.	

MQMD-Einstellungen in Befehlsnachrichten für den Warteschlangenmanager

Anwendungen, die Befehlsnachrichten an den Warteschlangenmanager senden, verwenden die folgenden Einstellungen für Felder im Nachrichtendeskriptor (MQMD). Felder, für die der Standardwert beibehalten oder für die wie gewohnt jeder gültige Wert festgelegt werden kann, sind hier nicht aufgeführt.

Bericht
Siehe `MsgType` und `CorrelId`.

MsgType
`MsgType` sollte auf *MQMT_REQUEST* oder *MQMT_DATAGRAM* gesetzt werden. *MQRC_MSG_TYPE_ERROR* wird zurückgegeben, falls `MsgType` nicht auf einen dieser Werte gesetzt ist.

MsgType sollte auf *MQMT_REQUEST* für eine Befehlsnachricht gesetzt werden, wenn immer eine Antwort erforderlich ist. Die Kennungen MQRO_PAN und MQRO_NAN im Feld Report haben in diesem Fall keine Bedeutung.

Wenn MsgType auf *MQMT_DATAGRAM* gesetzt wird, hängt es von den Einstellungen der Kennungen MQRO_PAN und MQRO_NAN im Feld Report ab, ob Antworten gesendet werden:

- Wenn nur MQRO_PAN gesetzt ist, sendet der Warteschlangenmanager nur dann eine Antwort, wenn der Befehl erfolgreich ausgeführt wird.
- Wenn nur MQRO_NAN gesetzt ist, sendet der Warteschlangenmanager nur dann eine Antwort, wenn der Befehl fehlschlägt.
- Wenn ein Befehl mit einer Warnung beendet wird, wird eine Antwort gesendet, sofern entweder MQRO_PAN oder MQRO_NAN gesetzt sind.
- Wenn MQRO_PAN und MQRO_NAN gesetzt sind, sendet der Warteschlangenmanager in jedem Fall eine Antwort, unabhängig davon, ob der Befehl erfolgreich ist oder fehlschlägt. Für den Warteschlangenmanager hat dies dieselbe Bedeutung, wie wenn die Einstellung MsgType auf MQMT_REQUEST gesetzt wird.
- Wenn weder MQRO_PAN noch MQRO_NAN gesetzt ist, wird in keinem Fall eine Antwort gesendet.

Format
Wird auf MQFMT_RF_HEADER_2 gesetzt

MsgId
Dieses Feld wird normalerweise auf MQMI_NONE gesetzt, damit der Warteschlangenmanager einen eindeutigen Wert generiert.

CorrelId
Für dieses Feld kann jeder beliebige Wert festgelegt werden. Wenn die Identität des Senders die Option CorrelId enthält, geben Sie diesen Wert zusammen mit MQRO_PASS_CORREL_ID im Feld Report an, um sicherzustellen, dass er in allen Antwortnachrichten festgelegt wird, die vom Warteschlangenmanager an den Sender übermittelt werden.

ReplyToQ
Dieses Feld definiert die Warteschlange, an die Antworten gesendet werden sollen, sofern überhaupt welche gesendet werden. Hierbei kann es sich um die Warteschlange des Senders handeln, was den Vorteil hat, dass der Parameter QName in der Nachricht weggelassen werden kann. Wenn Antworten jedoch an eine andere Warteschlange gesendet werden sollen, ist der Parameter QName erforderlich.

ReplyToQMgr
Dieses Feld definiert den Warteschlangenmanager für Antworten. Wenn Sie dieses Feld leer lassen (Standardwert), gibt der lokale Warteschlangenmanager dort seinen eigenen Namen an.

MQMD-Einstellungen für Veröffentlichungen, die von einem Warteschlangenmanager weitergeleitet wurden

Ein Warteschlangenmanager verwendet diese Einstellungen von Feldern im Nachrichtendeskriptor (MQMD), wenn er eine Veröffentlichung an einen Subskribenten sendet. Für alle anderen Felder im MQMD werden die Standardwerte verwendet.

Bericht
Report wird auf MQRO_NONE gesetzt.

MsgType
MsgType wird auf MQMT_DATAGRAM gesetzt.

Expiry
Expiry wird auf den Wert in der Nachricht Publish gesetzt, die von der Veröffentlichungskomponente empfangen wurde. Bei Nachrichten einer ständigen Veröffentlichung wird die noch ausstehende Zeitspanne um ungefähr die Dauer verringert, für die die Nachricht im Warteschlangenmanager verblieben ist.

Format
Format wird auf MQFMT_RF_HEADER_2 gesetzt.

MsgId
MsgId wird auf einen eindeutigen Wert gesetzt.

CorrelId
Wenn CorrelId Teil der Subskribentenidentität ist, wurde dieser Wert vom Subskribenten bei der Registrierung angegeben. Anderenfalls ist es ein Wert ungleich null, der vom Warteschlangenmanager festgelegt wurde.

Priorität
Priority nimmt den Wert an, der von der Veröffentlichungskomponente oder als aufgelöster Wert festgelegt wurde, wenn die Veröffentlichungskomponente MQPRI_PRIORITY_AS_Q_DEF angegeben hat.

Persistenz
Persistence nimmt den Wert an, der von der Veröffentlichungskomponente oder als aufgelöster Wert festgelegt wurde, wenn die Veröffentlichungskomponente MQPER_PERSISTENCE_AS_Q_DEF angegeben hat, sofern in der Nachricht Register Subscriber für den Subskribenten, an den diese Veröffentlichung gesendet wird, nichts anderes angegeben wurde.

ReplyToQ
ReplyToQ wird auf Leerzeichen gesetzt.

ReplyToQMgr
ReplyToQMgr wird auf den Namen des Warteschlangenmanagers gesetzt.

UserIdentifier
UserIdentifier ist die Benutzer-ID des Subskribenten, die dem bei der Registrierung des Subskribenten festgelegten Wert entspricht.

AccountingToken
AccountingToken ist das Abrechnungstoken des Subskribenten, das dem bei der ersten Registrierung des Subskribenten festgelegten Wert entspricht.

ApplIdentityData
ApplIdentityData sind die Anwendungsidentitätsdaten des Subskribenten, die dem bei der ersten Registrierung des Subskribenten festgelegten Wert entsprechen.

PutApplType
PutApplType wird auf MQAT_BROKER gesetzt.

PutApplName
PutApplName wird auf die ersten 28 Zeichen des Namens des Warteschlangenmanagers gesetzt.

PutDate
PutDate ist das Datum, an dem die Nachricht eingereiht wurde.

PutTime
PutTime ist die Uhrzeit, zu der die Nachricht eingereiht wurde.

ApplOriginData
ApplOriginData wird auf Leerzeichen gesetzt.

MQMD-Einstellungen in Antwortnachrichten des Warteschlangenmanagers

Ein Warteschlangenmanager verwendet diese Einstellungen von Feldern im Nachrichtendeskriptor (MQMD), wenn er eine Antwort auf eine Veröffentlichungsnachricht sendet. Für alle anderen Felder im MQMD werden die Standardwerte verwendet.

Bericht
Report wird auf null gesetzt.

MsgType
MsgType wird auf MQMT_REPLY gesetzt.

Format
Format wird auf MQFMT_RF_HEADER_2 gesetzt.

MsgId
Die Einstellung von MsgId hängt von den Optionen für den Parameter Report in der ursprünglichen Befehlsnachricht ab. Dieser Wert wird standardmäßig auf MQMI_NONE gesetzt, damit der Warteschlangenmanager einen eindeutigen Wert generiert.

CorrelId
Die Einstellung von CorrelId hängt von den Optionen für den Parameter Report in der ursprünglichen Befehlsnachricht ab. Dies bedeutet, dass für den Parameter CorrelId standardmäßig derselbe Wert wie für den Parameter MsgId der Befehlsnachricht festgelegt wird. Dies kann dazu verwendet werden, eine Korrelation zwischen den Befehlen und ihren Antworten herzustellen.

Priorität
Priority wird auf den Wert gesetzt, der in der ursprünglichen Befehlsnachricht angegeben wurde.

Persistenz
Persistence wird auf den Wert gesetzt, der in der ursprünglichen Befehlsnachricht angegeben wurde.

Expiry
Expiry wird auf den gleichen Wert wie in der ursprünglichen Befehlsnachricht gesetzt, die vom Warteschlangenmanager empfangen wurde.

PutApplType
PutApplType wird auf MQAT_BROKER gesetzt.

PutApplName
PutApplName wird auf die ersten 28 Zeichen des Warteschlangenmanagernamens gesetzt.

Andere Kontextfelder werden so festgelegt, als wären sie mit MQPMO_PASS_IDENTITY_CONTEXT generiert worden.

Maschinencodierungen

In diesem Abschnitt wird die Struktur des Felds *Encoding* im Nachrichtendeskriptor beschrieben.

„MQMD - Nachrichtendeskriptor" auf Seite 399 enthält eine Zusammenfassung der Felder in der Struktur.

Das Feld *Encoding* ist eine 32-Bit-Ganzzahl, die aus vier separaten Teilfeldern besteht, die folgende Angaben enthalten:

- Codierung für binäre Ganzzahlen
- Codierung für gepackt dezimale Ganzzahlen
- Codierung für Gleitkommazahlen
- Reservierte Bits

Jedes Teilfeld wird durch eine Bitmaske identifiziert, die an den Positionen, die dem Teilfeld entsprechen, Bits mit dem Wert 1 und an allen übrigen Positionen Bits mit dem Wert 0 enthält. Die Bits sind so nummeriert, dass das Bit 0 das höchstwertige Bit darstellt, während das Bit 31 das niedrigstwertige Bit ist. Folgende Masken sind definiert:

MQENC_INTEGER_MASK
Maske für die Codierung von binären Ganzzahlen.

Dieses Teilfeld belegt die Bitpositionen 28 bis 31 im Feld *Encoding*.

MQENC_DECIMAL_MASK
Maske für die Codierung von Ganzzahlen im gepackten Dezimalformat.

Dieses Teilfeld belegt die Bitpositionen 24 bis 27 im Feld *Encoding*.

MQENC_FLOAT_MASK
Maske für die Gleitkommacodierung.

Dieses Teilfeld belegt die Bitpositionen 20 bis 23 im Feld *Encoding*.

MQENC_RESERVED_MASK
Maske für reservierte Bits.

Dieses Teilfeld belegt die Bitpositionen 0 bis 19 im Feld *Encoding*.

Codierung von binären Ganzzahlen

Die folgenden Werte sind für die Codierung von binären Ganzzahlen gültig:

MQENC_INTEGER_UNDEFINED
Binäre Ganzzahlen werden mit einer nicht definierten Codierung dargestellt.

MQENC_INTEGER_NORMAL
Binäre Ganzzahlen werden auf die herkömmliche Art und Weise dargestellt:

- Das niedrigstwertige Byte in der Zahl hat die höchste Adresse aller Bytes in der Zahl; das höchstwertige Byte hat die niedrigste Adresse
- Das niedrigstwertige Bit in jedem Byte grenzt an das Byte mit der nächsthöheren Adresse; das höchstwertige Bit in jedem Byte grenzt an das Byte mit der nächstniedrigeren Adresse

MQENC_INTEGER_REVERSED
Binäre Ganzzahlen werden genauso dargestellt wie MQENC_INTEGER_NORMAL, dabei sind die Bytes jedoch in umgekehrter Reihenfolge angeordnet. Die Bits in jedem Byte sind genauso angeordnet wie MQENC_INTEGER_NORMAL.

Codierung für gepackt dezimale Ganzzahlen

Die folgenden Werte sind für die Codierung von Ganzzahlen im gepackten Dezimalformat gültig:

MQENC_DECIMAL_UNDEFINED
Ganzzahlen im gepackten Dezimalformat werden mit einer nicht definierten Codierung dargestellt.

MQENC_DECIMAL_NORMAL
Ganzzahlen im gepackten Dezimalformat werden auf die herkömmliche Art und Weise dargestellt:

- Jede Dezimalziffer in der druckbaren Form der Zahl wird gepackt dezimal durch eine einzige Hexadezimalziffer im Bereich von X'0' bis X'9' dargestellt. Jede Hexadezimalziffer belegt vier Bits, und somit stellt jedes Byte in der gepackten Dezimalzahl zwei Dezimalziffern in der druckbaren Form der Zahl dar.
- Das niedrigstwertige Byte in der gepackten Dezimalzahl ist das Byte, das die niedrigstwertige Dezimalziffer enthält. Innerhalb dieses Bytes enthalten die höchstwertigen vier Bits die niedrigstwertige Dezimalziffer und die niedrigstwertigen vier Bits enthalten das Vorzeichen. Das Vorzeichen ist X'C' (positiv), X'D' (negativ) oder X'F' (ohne Vorzeichen).
- Das niedrigstwertige Byte in der Zahl hat die höchste Adresse aller Bytes in der Zahl; das höchstwertige Byte hat die niedrigste Adresse
- Das niedrigstwertige Bit in jedem Byte grenzt an das Byte mit der nächsthöheren Adresse; das höchstwertige Bit in jedem Byte grenzt an das Byte mit der nächstniedrigeren Adresse.

MQENC_DECIMAL_REVERSED
Ganzzahlen im gepackten Dezimalformat werden genauso dargestellt wie MQENC_DECIMAL_NORMAL, dabei sind die Bytes jedoch in umgekehrter Reihenfolge angeordnet. Die Bits in jedem Byte sind genauso angeordnet wie MQENC_DECIMAL_NORMAL.

Gleitkommacodierung

Die folgenden Werte sind für die Codierung von Gleitkommazahlen gültig:

MQENC_FLOAT_UNDEFINED
Gleitkommazahlen werden mit einer nicht definierten Codierung dargestellt.

MQENC_FLOAT_IEEE_NORMAL
Gleitkommazahlen werden mit dem Standard IEEE dargestellt.[3]Gleitkommaformat, wobei die Bytes wie folgt angeordnet sind:

- Das niedrigstwertige Byte in der Mantisse hat die höchste Adresse aller Bytes in der Zahl; das Byte mit dem Exponenten hat die niedrigste Adresse
- Das niedrigstwertige Bit in jedem Byte grenzt an das Byte mit der nächsthöheren Adresse; das höchstwertige Bit in jedem Byte grenzt an das Byte mit der nächstniedrigeren Adresse

Einzelheiten zur IEEE-Codierung von Gleitkommazahlen finden Sie in IEEE-Standard 754.

MQENC_FLOAT_IEEE_REVERSED
Gleitkommazahlen werden genauso dargestellt wie MQENC_FLOAT_IEEE_NORMAL, dabei sind die Bytes jedoch in umgekehrter Reihenfolge angeordnet. Die Bits in jedem Byte sind genauso angeordnet wie MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390
Gleitkommazahlen werden mit dem System/390-Standardformat für Gleitkommazahlen dargestellt. Dieses Format wird auch von System/370 verwendet.

Codierungen erstellen

Um einen Wert für das Feld *Encoding* in MQMD zu erstellen, können die relevanten Konstanten, die die erforderlichen Codierungen beschreiben:

- Zusammen hinzugefügt werden oder
- Mit der bitweisen ODER-Operation kombiniert werden (wenn die Programmiersprache Bitoperationen unterstützt)

Kombinieren Sie unabhängig von der verwendeten Methode nur eine der MQENC_INTEGER_*-Codierungen mit einer der MQENC_DECIMAL_*-Codierungen und einer der MQENC_FLOAT_*-Codierungen.

Codierungen analysieren

Das Feld *Encoding* enthält Unterfelder. Daher müssen Anwendungen, die die Codierung von Ganzzahlen, gepackten Dezimalzahlen oder Gleitkommazahlen prüfen müssen, eines der beschriebenen Verfahren verwenden.

Verwenden von Bitoperationen

Wenn die Programmiersprache Bitoperationen unterstützt, führen Sie einen der folgenden Schritte aus:

1. Wählen Sie je nach dem Typ der erforderlichen Codierung einen der folgenden Werte aus:
   - MQENC_INTEGER_MASK für die Codierung von binären Ganzzahlen
   - MQENC_DECIMAL_MASK für die Codierung von Ganzzahlen im gepackten Dezimalformat
   - MQENC_FLOAT_MASK für die Codierung von Gleitkommazahlen

   Rufen Sie den Wert A auf.
2. Kombinieren Sie das Feld *Encoding* mit A unter Verwendung der bitweisen UND-Operation. Rufen Sie das Ergebnis B auf.
3. B ist die erforderliche Codierung und kann auf die Gleichheit mit jedem der Werte getestet werden, die für diesen Typ der Codierung gültig sind.

Verwenden von Arithmetik

Wenn die Programmiersprache Bitoperationen *nicht* unterstützt, führen Sie die folgenden Schritte mithilfe von Ganzzahlarithmetik aus:

1. Wählen Sie je nach dem Typ der erforderlichen Codierung einen der folgenden Werte aus:
   - 1 für die Codierung von binären Ganzzahlen
   - 16 für die Codierung von Ganzzahlen im gepackten Dezimalformat

[3] Das Institute of Electrical and Electronics Engineers

- 256 für die Codierung von Gleitkommazahlen

Rufen Sie den Wert A auf.

2. Dividieren Sie den Wert im Feld *Encoding* durch A. Rufen Sie das Ergebnis B auf.
3. Dividieren Sie B durch 16. Nennen Sie das Ergebnis C.
4. Multiplizieren Sie C mit 16 und subtrahieren Sie den Wert von B; rufen Sie das Ergebnis D auf.
5. Multiplizieren Sie D mit A; rufen Sie das Ergebnis E auf.
6. E ist die erforderliche Codierung und kann auf die Gleichheit mit jedem der Werte getestet werden, die für diesen Typ der Codierung gültig sind.

Zusammenfassung der Maschinenarchitektur-Codierungen

Zu den Codierungen für Maschinenarchitekturen siehe Tabelle 578 auf Seite 904.

Tabelle 578. Zusammenfassung der Codierungen für Maschinenarchitekturen

Maschinenarchitektur	Codierung von binären Ganzzahlen	Codierung von Ganzzahlen im gepackten Dezimalformat	Gleitkommacodierung
IBM i	normal	normal	IEEE normal
Intel x86	umgekehrt	umgekehrt	IEEE umgekehrt
PowerPC	normal	normal	IEEE normal
System/390	normal	normal	System/390

Report options and message flags

In diesem Abschnitt werden die Felder *Report* und *MsgFlags* beschrieben, die Teil des Nachrichtendeskriptors MQMD sind, der in den MQGET-, MQPUT- und MQPUT1-Aufrufen angegeben wird.

In diesem Abschnitt werden folgende Themen behandelt:

- Struktur des Berichtsfelds und Verarbeitungsweise des Warteschlangenmanagers
- Analyse des Berichtsfeldes durch eine Anwendung
- Struktur des Feldes für Nachrichtenflags

Weitere Informationen zum Nachrichtendeskriptor MQMD finden Sie im Abschnitt „MQMD - Nachrichtendeskriptor” auf Seite 399.

Struktur des Berichtsfelds

Diese Informationen beschreiben die Struktur des Berichtsfeldes.

Das Feld *Report* ist eine 32-Bit-Ganzzahl, die in drei separate Unterfelder unterteilt wird. Diese Unterfelder identifizieren Folgendes:

- Berichtsoptionen, die abgelehnt werden, wenn der lokale Warteschlangenmanager sie nicht erkennt
- Berichtsoptionen, die immer akzeptiert werden, auch wenn der lokale Warteschlangenmanager sie nicht erkennt
- Berichtsoptionen, die nur akzeptiert werden, wenn bestimmte andere Bedingungen erfüllt sind

Jedes Teilfeld wird durch eine Bitmaske identifiziert, die an den Positionen, die dem Teilfeld entsprechen, Bits mit dem Wert 1 und an allen übrigen Positionen Bits mit dem Wert 0 enthält. Die Bits in einem Unterfeld grenzen nicht unbedingt aneinander. Die Bits sind so nummeriert, dass das Bit 0 das höchstwertige Bit darstellt, während das Bit 31 das niedrigstwertige Bit ist. Die folgenden Masken sind definiert, um die Unterfelder zu identifizieren:

MQRO_REJECT_UNSUP_MASK
Diese Maske gibt die Bitpositionen im Feld *Report* an, wobei Berichtsoptionen, die nicht vom lokalen Warteschlangenmanager unterstützt werden, dazu führen, dass der MQPUT- oder MQPUT1-Aufruf mit dem Beendigungscode MQCC_FAILED und dem Ursachencode MQRC_REPORT_OPTIONS_ERROR fehlschlägt.

Dieses Unterfeld belegt die Bitpositionen 3 und 11 bis 13.

MQRO_ACCEPT_UNSUP_MASK
Diese Maske gibt die Bitpositionen im Feld *Report* an, wobei Berichtsoptionen, die nicht vom lokalen Warteschlangenmanager unterstützt werden, trotzdem in den MQPUT- oder MQPUT1-Aufrufen akzeptiert werden. In diesem Fall wird der Beendigungscode MQCC_WARNING mit dem Ursachencode MQRC_UNKNOWN_REPORT_OPTION zurückgegeben.

Dieses Unterfeld belegt die Bitpositionen 0 bis 2, 4 bis 10 und 24 bis 31.

Die folgenden Berichtsoptionen sind in diesem Unterfeld enthalten:

- 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
Diese Maske gibt die Bitpositionen im Feld *Report* an, wobei Berichtsoptionen, die nicht vom lokalen Warteschlangenmanager unterstützt werden, trotzdem in den MQPUT- oder MQPUT1-Aufrufen unterstützt werden, *wenn* beide der folgenden Bedingungen erfüllt sind:

- Die Nachricht ist für einen fernen Warteschlangenmanager bestimmt.
- Die Anwendung reiht die Nachricht nicht direkt in einer lokalen Übertragungswarteschlange ein (d. h. die durch die Felder *ObjectQMgrName* und *ObjectName* angegebene Warteschlange im Objektdeskriptor, der im MQOPEN- oder MQPUT1-Aufruf angegeben ist, ist keine lokale Übertragungswarteschlange).

Wenn diese Bedingungen erfüllt sind, wird der Beendigungscode MQCC_WARNING mit dem Ursachencode MQRC_UNKNOWN_REPORT_OPTION zurückgegeben. Sind die Bedingungen nicht erfüllt, wird MQCC_FAILED mit dem Ursachencode MQRC_REPORT_OPTIONS_ERROR zurückgegeben.

Dieses Unterfeld belegt die Bitpositionen 14 bis 23.

Die folgenden Berichtsoptionen sind in diesem Unterfeld enthalten:

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA

- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Wenn im Feld *Report* Optionen angegeben sind, die der Warteschlangenmanager nicht erkennt, prüft der Warteschlangenmanager jedes Unterfeld mit der bitweisen UND-Operation, um das Feld *Report* mit der Maske für dieses Unterfeld zu kombinieren. Ist das Ergebnis dieser Operation nicht null, werden die oben genannten Beendigungscodes und Ursachencodes zurückgegeben.

Wenn MQCC_WARNING zurückgegeben wird, ist nicht definiert, welcher Ursachencode zurückgegeben wird, wenn andere Warnbedingungen vorhanden sind.

Die Möglichkeit zum Angeben und Akzeptieren von Berichtsoptionen, die nicht vom lokalen Warteschlangenmanager erkannt werden, ist hilfreich, wenn eine Nachricht mit einer Berichtsoption gesendet wird, die von einem *fernen* Warteschlangenmanager erkannt und verarbeitet wird.

Analysieren des Berichtsfelds

Das Feld *Report* enthält Unterfelder: Daher müssen Anwendungen, die prüfen müssen, ob der Absender der Nachricht einen bestimmten Bericht angefordert hat, eines der beschriebenen Verfahren verwenden.

Verwenden von Bitoperationen

Wenn die Programmiersprache Bitoperationen unterstützt, führen Sie einen der folgenden Schritte aus:

1. Wählen Sie je nach dem Typ des zu prüfenden Berichts einen der folgenden Werte aus:
   - MQRO_COA_WITH_FULL_DATA für COA-Bericht
   - MQRO_COD_WITH_FULL_DATA für COD-Bericht
   - MQRO_EXCEPTION_WITH_FULL_DATA für Abweichungsbericht
   - MQRO_EXPIRATION_WITH_FULL_DATA für Ablaufbericht

   Rufen Sie den Wert A auf.

   Unter z/OS verwenden Sie die MQRO_*_WITH_DATA-Werte anstelle der MQRO_*_WITH_FULL_DATA-Werte.
2. Kombinieren Sie das Feld *Report* mit A unter Verwendung der bitweisen UND-Operation. Rufen Sie das Ergebnis B auf.
3. Testen Sie B auf die Gleichheit mit jedem Wert, der für diesen Berichtstyp möglich ist.

   Beispiel: Wenn A MQRO_EXCEPTION_WITH_FULL_DATA ist, testen Sie B auf die Gleichheit mit jedem der folgenden Werte, um zu ermitteln, was vom Absender der Nachricht angegeben wurde:
   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Die Prüfungen können in der Reihenfolge durchgeführt werden, die für die Anwendungslogik am zweckmäßigsten ist.

Verwenden Sie eine ähnliche Methode zum Testen der Option MQRO_PASS_MSG_ID oder MQRO_PASS_CORREL_ID. Wählen Sie für den Wert A die geeignete dieser beiden Konstanten aus und fahren Sie dann wie oben beschrieben fort.

Verwenden von Arithmetik

Wenn die Programmiersprache Bitoperationen *nicht* unterstützt, führen Sie die folgenden Schritte mithilfe von Ganzzahlarithmetik aus:

1. Wählen Sie je nach dem Typ des zu prüfenden Berichts einen der folgenden Werte aus:

- MQRO_COA für COA-Bericht
- MQRO_COD für COD-Bericht
- MQRO_EXCEPTION für Abweichungsbericht
- MQRO_EXPIRATION für Ablaufbericht

Rufen Sie den Wert A auf.

2. Dividieren Sie den Wert im Feld *Report* durch A. Rufen Sie das Ergebnis B auf.
3. Dividieren Sie B durch 8; rufen Sie das Ergebnis C auf.
4. Multiplizieren Sie C mit 8 und subtrahieren Sie den Wert von B; rufen Sie das Ergebnis D auf.
5. Multiplizieren Sie D mit A; rufen Sie das Ergebnis E auf.
6. Testen Sie E auf die Gleichheit mit jedem Wert, der für diesen Berichtstyp möglich ist.

   Beispiel: Wenn A MQRO_EXCEPTION ist, testen Sie E auf die Gleichheit mit jedem der folgenden Werte, um zu ermitteln, was vom Absender der Nachricht angegeben wurde:

   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Die Prüfungen können in der Reihenfolge durchgeführt werden, die für die Anwendungslogik am zweckmäßigsten ist.

Der folgende Pseudocode veranschaulicht dieses Verfahren für Ausnahmebericht-Nachrichten:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Verwenden Sie eine ähnliche Methode zum Testen der Option MQRO_PASS_MSG_ID oder MQRO_PASS_CORREL_ID. Wählen Sie für den Wert A die geeignete dieser beiden Konstanten aus und fahren Sie dann wie oben beschrieben fort. Ersetzen Sie dabei jedoch den Wert 8 in den oben genannten Schritten durch den Wert 2.

Struktur des Felds für Nachrichtenflags

Diese Informationen beschreiben die Struktur des Felds für Nachrichtenflags.

Das Feld *MsgFlags* ist eine 32-Bit-Ganzzahl, die in drei separate Unterfelder unterteilt wird. Diese Unterfelder identifizieren Folgendes:

- Nachrichtenflags, die abgelehnt werden, wenn der lokale Warteschlangenmanager sie nicht erkennt
- Nachrichtenflags, die immer akzeptiert werden, auch wenn der lokale Warteschlangenmanager sie nicht erkennt
- Nachrichtenflags, die nur akzeptiert werden, wenn bestimmte andere Bedingungen erfüllt sind

Anmerkung: Alle Unterfelder in *MsgFlags* sind für die Verwendung durch den Warteschlangenmanager reserviert.

Jedes Teilfeld wird durch eine Bitmaske identifiziert, die an den Positionen, die dem Teilfeld entsprechen, Bits mit dem Wert 1 und an allen übrigen Positionen Bits mit dem Wert 0 enthält. Die Bits sind so nummeriert, dass das Bit 0 das höchstwertige Bit darstellt, während das Bit 31 das niedrigstwertige Bit ist. Die folgenden Masken sind definiert, um die Unterfelder zu identifizieren:

MQMF_REJECT_UNSUP_MASK
Diese Maske gibt die Bitpositionen im Feld *MsgFlags* an, wobei Nachrichtenflags, die nicht vom lokalen Warteschlangenmanager unterstützt werden, dazu führen, dass der MQPUT- oder MQPUT1-

Aufruf mit dem Beendigungscode MQCC_FAILED und dem Ursachencode MQRC_MSG_FLAGS_ERROR fehlschlägt.

Dieses Unterfeld belegt die Bitpositionen 20 bis 31.

Die folgenden Nachrichtenflags sind in diesem Unterfeld enthalten:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBITED

MQMF_ACCEPT_UNSUP_MASK
Diese Maske gibt die Bitpositionen im Feld *MsgFlags* an, wobei Nachrichtenflags, die nicht vom lokalen Warteschlangenmanager unterstützt werden, trotzdem in den MQPUT- oder MQPUT1-Aufrufen akzeptiert werden. Der Beendigungscode lautet MQCC_OK.

Dieses Unterfeld belegt die Bitpositionen 0 bis 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK
Diese Maske gibt die Bitpositionen im Feld *MsgFlags* an, wobei Nachrichtenflags, die nicht vom lokalen Warteschlangenmanager unterstützt werden, trotzdem in den MQPUT- oder MQPUT1-Aufrufen unterstützt werden, *wenn* beide der folgenden Bedingungen erfüllt sind:

- Die Nachricht ist für einen fernen Warteschlangenmanager bestimmt.
- Die Anwendung reiht die Nachricht nicht direkt in einer lokalen Übertragungswarteschlange ein (d. h. die durch die Felder *ObjectQMgrName* und *ObjectName* angegebene Warteschlange im Objektdeskriptor, der im MQOPEN- oder MQPUT1-Aufruf angegeben ist, ist keine lokale Übertragungswarteschlange).

Wenn diese Bedingungen erfüllt sind, wird der Beendigungscode MQCC_OK zurückgegeben. Sind die Bedingungen nicht erfüllt, wird MQCC_FAILED mit dem Ursachencode MQRC_MSG_FLAGS_ERROR zurückgegeben.

Dieses Unterfeld belegt die Bitpositionen 12 bis 19.

Wenn im Feld *MsgFlags* Flags angegeben sind, die der Warteschlangenmanager nicht erkennt, prüft der Warteschlangenmanager jedes Unterfeld mit der bitweisen UND-Operation, um das Feld *MsgFlags* mit der Maske für dieses Unterfeld zu kombinieren. Ist das Ergebnis dieser Operation nicht null, werden die oben genannten Beendigungscodes und Ursachencodes zurückgegeben.

Datenkonvertierungsexit

In dieser Themensammlung wird die Schnittstelle zum Datenkonvertierungsexit sowie die Verarbeitung beschrieben, die der Warteschlangenmanager durchführt, wenn eine Datenkonvertierung erforderlich ist.

Weitere Informationen zur Datenkonvertierung finden Sie im Abschnitt *Datenkonvertierung unter WebSphere MQ* unter https://www.ibm.com/support/docview.wss?uid=swg27005729.

Der Datenkonvertierungsexit wird im Rahmen der Verarbeitung des MQGET-Aufrufs aufgerufen, um die Anwendungsnachrichtendaten in die Darstellung zu konvertieren, die von der empfangenden Anwendung angefordert wird. Die Konvertierung der Anwendungsnachrichtendaten ist optional. Damit sie durchgeführt wird, muss im MQGET-Aufruf die Option MQGMO_CONVERT angegeben werden.

Folgende Themen werden behandelt:

- Verarbeitung durch den Warteschlangenmanager als Reaktion auf die Option MQGMO_CONVERT (siehe „Konvertierungsverarbeitung” auf Seite 909)

- Verarbeitungskonventionen, die der Warteschlangenmanager bei der Verarbeitung eines integrierten Formats beachtet und die auch für vom Benutzer geschriebene Exits empfohlen werden Siehe hierzu „Verarbeitungskonventionen" auf Seite 910.
- Besondere Hinweise zur Konvertierung von Berichtsnachrichten (siehe „Konvertierung von Berichtsnachrichten" auf Seite 914)
- Parameter, die an den Datenkonvertierungsexit übergeben werden (siehe „MQ_DATA_CONV_EXIT - Datenkonvertierungsexit" auf Seite 927).
- Ein Aufruf, der im Exit zur Konvertierung von Zeichendaten zwischen verschiedenen Darstellungen verwendet werden kann (siehe „MQXCNVC – Zeichen konvertieren" auf Seite 921)
- Datenstrukturparameter, der für den Exit spezifisch ist (siehe „MQDXP - Parameter des Datenkonvertierungsexits" auf Seite 915)

Konvertierungsverarbeitung

Diese Informationen beschreiben die Verarbeitung durch den Warteschlangenmanager als Reaktion auf die Option MQGMO_CONVERT.

Der Warteschlangenmanager führt die folgenden Aktionen aus, wenn die Option MQGMO_CONVERT im MQGET-Aufruf angegeben wird und eine Nachricht vorliegt, die an die Anwendung zurückgegeben werden soll:

1. Wenn mindestens eine der folgenden Bedingungen erfüllt ist, ist keine Konvertierung erforderlich:
   - Die Nachrichtendaten sind bereits im Zeichensatz und der Codierung enthalten, die von der Anwendung angefordert werden, welche den MQGET-Aufruf ausgibt. Die Anwendung muss die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* des MQGET-Aufrufs auf die erforderlichen Werte setzen, bevor sie den Aufruf ausgibt.
   - Die Länge der Nachrichtendaten ist gleich Null.
   - Die Länge des Parameters *Buffer* für den MQGET-Aufruf ist gleich Null.

   In diesen Fällen wird die Nachricht ohne Konvertierung an die Anwendung zurückgegeben, die den MQGET-Aufruf ausgibt. Die Werte *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* werden auf die Werte in den Steuerinformationen der Nachricht gesetzt und der Aufruf wird mit einer der folgenden Kombinationen aus Beendigungscode und Ursachencode ausgeführt:

Beendigungscode	Ursachencode
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_WARNING	MQRC_TRUNCATED_MSG_FAILED

   Die folgenden Schritte werden nur ausgeführt, wenn der Zeichensatz oder die Codierung der Nachrichtendaten vom entsprechenden Wert im Parameter *MsgDesc* abweichen und zu konvertierende Daten vorliegen:
2. Wenn das Feld *Format* in den Steuerinformationen der Nachricht den Wert MQFMT_NONE aufweist, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_FORMAT_ERROR zurückgegeben.

   In allen anderen Fällen wird die Konvertierungsverarbeitung fortgesetzt.
3. Die Nachricht wird aus der Warteschlange entfernt und in einem temporären Puffer abgelegt, der dieselbe Größe hat wie der Parameter *Buffer*. Für Suchoperationen wird die Nachricht in den temporären Puffer kopiert und nicht aus der Warteschlange entfernt.
4. Wenn die Nachricht abgeschnitten werden muss, damit sie in den Puffer passt, werden die folgenden Schritte ausgeführt:
   - Wenn die Option MQGMO_ACCEPT_TRUNCATED_MSG *nicht* angegeben wurde, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_FAILED zurückgegeben.

- Wenn die Option MQGMO_ACCEPT_TRUNCATED_MSG *angegeben* wurde, wird der Beendigungscode auf MQCC_WARNING gesetzt. Der Ursachencode wird auf MQRC_TRUNCATED_MSG_ACCEPTED gesetzt und die Konvertierungsverarbeitung wird fortgesetzt.

5. Wenn die Nachricht ohne Abschneiden im Puffer abgelegt werden kann oder die Option MQGMO_ACCEPT_TRUNCATED_MSG angegeben wurde, werden die folgenden Schritte ausgeführt:
   - Wenn das Format ein integriertes Format ist, wird der Puffer an den Datenkonvertierungsservice des Warteschlangenmanagers übergeben.
   - Ist das Format kein integriertes Format, wird der Puffer an einen benutzerdefinierten Exit übergeben, der denselben Namen hat wie das Format. Wenn der Exit nicht gefunden werden kann, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_FORMAT_ERROR zurückgegeben.

   Wenn kein Fehler auftritt, ist die Ausgabe aus dem Datenkonvertierungsservice oder dem benutzerdefinierten Exit die konvertierte Nachricht mit dem Beendigungscode und Ursachencode, die an die Anwendung zurückgegeben werden soll, die den MQGET-Aufruf ausgibt.
6. Wenn die Konvertierung erfolgreich ist, gibt der Warteschlangenmanager die konvertierte Nachricht an die Anwendung zurück. In diesem Fall weisen der vom MQGET-Aufruf zurückgegebene Beendigungscode und Ursachencode eine der folgenden Kombinationen auf:

Beendigungscode	**Ursachencode**
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED

   Wenn die Konvertierung durch einen benutzerdefinierten Exit ausgeführt wird, können jedoch andere Ursachencodes zurückgegeben werden, auch wenn die Konvertierung erfolgreich ist.

   Wenn die Konvertierung fehlschlägt, gibt der Warteschlangenmanager die konvertierte Nachricht an die Anwendung zurück. Dabei sind die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* auf die Werte in den Steuerinformationen der Nachricht gesetzt. Der Beendigungscode lautet MQCC_WARNING.

Verarbeitungskonventionen

Beim Konvertieren eines integrierten Formats folgt der Warteschlangenmanager den beschriebenen Verarbeitungskonventionen.

Benutzerdefinierte Exits sollten ebenfalls diese Konventionen befolgen, dies wird jedoch nicht vom Warteschlangenmanager durchgesetzt. Folgende integrierte Formate werden vom Warteschlangenmanager konvertiert:

- MQFMT_ADMIN
- MQFMT_CICS (nur z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT Version 1
- MQFMT_EVENT Version 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 (nur z/OS)
- MQFMT_XMIT_Q_HEADER

1. Wenn die Nachricht während der Konvertierung erweitert wird und die Größe des Parameters *Buffer* überschreitet, werden folgende Schritte ausgeführt:
   - Wenn die Option MQGMO_ACCEPT_TRUNCATED_MSG *nicht* angegeben wurde, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_CONVERTED_MSG_TOO_BIG zurückgegeben.
   - Wenn die Option MQGMO_ACCEPT_TRUNCATED_MSG *angegeben* wurde, wird die Nachricht abgeschnitten und der Beendigungscode wird auf MQCC_WARNING gesetzt. Der Ursachencode wird auf MQRC_TRUNCATED_MSG_ACCEPTED gesetzt und die Konvertierungsverarbeitung wird fortgesetzt.
2. Wenn (vor oder nach der Konvertierung) ein Abschneiden erfolgt, kann die Anzahl gültiger Bytes, die im Parameter *Buffer* zurückgegeben werden, *kleiner als* die Länge des Puffers sein.

   Dies kann z. B. eintreten, wenn eine 4-Byte-Ganzzahl oder ein Doppelbytezeichen über das Ende des Puffers hinausgeht. Das unvollständige Informationselement wird nicht konvertiert und die Bytes in der zurückgegebenen Nachricht enthalten keine gültigen Informationen. Dies kann auch der Fall sein, wenn eine Nachricht, die vor der Konvertierung abgeschnitten wurde, während der Konvertierung kleiner wird.

   Wenn die Anzahl gültiger zurückgegebener Byte kleiner ist als die Länge des Puffers, werden die nicht verwendeten Byte am Ende des Puffers auf Nullen gesetzt.
3. Wenn ein Array oder eine Zeichenfolge über das Ende des Puffers hinaus geht, werden so viele Daten wie möglich konvertiert. Lediglich das unvollständige Array-Element oder Doppelbytezeichen wird nicht konvertiert, vorausgehende Array-Elemente oder Zeichen werden konvertiert.
4. Wenn (vor oder nach der Konvertierung) ein Abschneiden erfolgt, ist die für den Parameter *DataLength* zurückgegebene Länge gleich der Länge der *unkonvertierten* Nachricht vor dem Abschneiden.
5. Wenn Zeichenfolgen zwischen Einzelbytezeichensätzen, Doppelbytezeichensätzen oder Mehrbytezeichensätzen konvertiert werden, können die Zeichenfolgen länger oder kürzer werden.
   - In den PCF-Formaten MQFMT_ADMIN, MQFMT_EVENT und MQFMT_PCF werden die Zeichenfolgen in der Struktur MQCFST und MQCFSL nach Bedarf verlängert oder verkürzt, um die Zeichenfolge nach der Konvertierung unterzubringen.

     Für die Zeichenfolgenlistenstruktur MQCFSL können die Zeichenfolgen in der Liste um unterschiedliche Beträge erweitert oder verkürzt werden. In diesem Fall füllt der Warteschlangenmanager die kürzeren Zeichenfolgen mit Leerzeichen auf, damit diese dieselbe Länge aufweisen wie die längste Zeichenfolge nach der Konvertierung.
   - Im Format MQFMT_REF_MSG_HEADER werden die durch die Felder *SrcEnvOffset*, *SrcNameOffset*, *DestEnvOffset* und *DestNameOffset* adressierten Zeichenfolgen nach Bedarf verlängert oder verkürzt, um die Zeichenfolgen nach der Konvertierung unterzubringen.
   - Im Format MQFMT_RF_HEADER wird das *NameValueString* nach Bedarf erweitert oder verkürzt, um die Name/Wert-Paare nach der Konvertierung unterzubringen.
   - In Strukturen mit festen Feldgrößen ermöglicht der Warteschlangenmanager, dass Zeichenfolgen innerhalb ihrer festen Felder erweitert oder verkürzt werden, sofern dabei keine kritischen Informationen verloren gehen. In dieser Hinsicht werden abschließende Leerzeichen und Zeichen, die auf das erste Nullzeichen im Feld folgen, als nicht relevant behandelt.
     - Wenn die Zeichenfolge erweitert wird, jedoch nur unkritische Zeichen gelöscht werden müssen, um die konvertierte Zeichenfolge im Feld unterzubringen, wird die Konvertierung erfolgreich ausgeführt und der Aufruf wird mit MQCC_OK und dem Ursachencode MQRC_NONE abgeschlossen (sofern keine anderen Fehler vorliegen).

– Wenn die Zeichenfolge erweitert wird, die konvertierte Zeichenfolge jedoch das Löschen kritischer Zeichen erfordert, damit sie in das Feld passt, wird die Nachricht unkonvertiert zurückgegeben und der Aufruf wird mit MQCC_WARNING und dem Ursachencode MQRC_CONVERTED_STRING_TOO_BIG abgeschlossen.

  Anmerkung: Der Ursachencode MQRC_CONVERTED_STRING_TOO_BIG wird in diesem Fall unabhängig davon ausgegeben, ob die Option MQGMO_ACCEPT_TRUNCATED_MSG angegeben wurde.

– Wenn die Zeichenfolge verkürzt wird, füllt der Warteschlangenmanager die Zeichenfolge mit Leerzeichen auf die Länge des Felds auf.

6. Für Nachrichten, die aus einer oder mehreren MQ-Headerstrukturen, gefolgt von Benutzerdaten bestehen, wird eine oder mehrere der Headerstrukturen möglicherweise konvertiert, während die übrige Nachricht nicht konvertiert wird. Abgesehen von zwei Ausnahmen geben die Felder *CodedCharSetId* und *Encoding* in jeder Headerstruktur jedoch immer den Zeichensatz und die Codierung der Daten, die auf die Headerstruktur folgen, korrekt an.

   Die beiden Ausnahmen sind die Strukturen MQCIH und MQIIH, bei denen die Werte in den Feldern *CodedCharSetId* und *Encoding* in diesen Strukturen nicht relevant sind. Für diese Strukturen befinden sich die Daten, die auf die Struktur folgen, im selben Zeichensatz und derselben Codierung wie die Struktur MQCIH oder MQIIH selbst.

7. Wenn die Felder *CodedCharSetId* oder *Encoding* in den Steuerinformationen der abgerufenen Nachricht oder im Parameter *MsgDesc* Werte angeben, die nicht definiert sind oder nicht unterstützt werden, ignoriert der Warteschlangenmanager unter Umständen den Fehler, wenn der nicht definierte oder nicht unterstützte Wert nicht beim Konvertieren der Nachricht verwendet werden muss.

   Beispiel: Wenn das Feld *Encoding* in der Nachricht eine nicht unterstützte Codierung von Gleitkommazahlen angibt, die Nachricht jedoch nur Ganzzahldaten enthält oder Gleitkommadaten enthält, die keine Konvertierung erfordern (da die Quell- und Zielcodierungen der Gleitkommazahlen identisch sind), wird der Fehler unter Umständen nicht diagnostiziert.

   Wenn der Fehler diagnostiziert wird, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und einem der Ursachencodes MQRC_SOURCE_*_ERROR oder MQRC_TARGET_*_ERROR zurückgegeben. Die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* werden auf die Werte in den Steuerinformationen der Nachricht gesetzt.

   Wenn der Fehler nicht diagnostiziert wird und die Konvertierung erfolgreich abgeschlossen wird, sind die in den Feldern *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* zurückgegebenen Werte die Werte, die in der Anwendung angegeben wurden, die den MQGET-Aufruf ausgibt.

8. Wenn die Nachricht unkonvertiert an die Anwendung zurückgegeben wird, wird der Beendigungscode in allen Fällen auf MQCC_WARNING gesetzt und die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* werden auf die Werte gesetzt, die sich für die unkonvertierten Daten eignen. Dies wird auch für MQFMT_NONE ausgeführt.

   Der Parameter *Reason* wird auf einen Code gesetzt, der angibt, warum die Konvertierung nicht ausgeführt wurde, sofern die Nachricht nicht abgeschnitten werden musste. Ursachencodes im Hinblick auf das Abschneiden haben Vorrang vor Ursachencodes im Zusammenhang mit der Konvertierung. (Um zu ermitteln, ob eine abgeschnittene Nachricht konvertiert wurde, prüfen Sie die in den Feldern *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* zurückgegebenen Werte.)

   Wenn ein Fehler diagnostiziert wird, wird entweder ein spezifischer Ursachencode zurückgegeben oder der allgemeine Ursachencode MQRC_NOT_CONVERTED. Welcher Ursachencode zurückgegeben wird, hängt von den Diagnosefunktionen des zugrunde liegenden Datenkonvertierungsservice ab.

9. Wenn der Beendigungscode MQCC_WARNING zurückgegeben wird und mehrere Ursachencodes relevant sind, gilt folgende Reihenfolge:

   a. Die folgenden Ursachencodes haben Vorrang vor allen anderen. Es kann jeweils nur eine der Ursachen in dieser Gruppe auftreten:

      • MQRC_SIGNAL_REQUEST_ACCEPTED
      • MQRC_TRUNCATED_MSG_ACCEPTED

b. Die Reihenfolge der verbleibenden Ursachencodes ist nicht definiert.

10. Bei Abschluss des MQGET-Aufrufs:
    - Der folgende Ursachencode gibt an, dass die Nachricht erfolgreich konvertiert wurde:
      - MQRC_NONE
    - Mit folgenden Ursachencodes wird angegeben, dass die Nachricht *möglicherweise* erfolgreich konvertiert wurde (prüfen Sie die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc*):
      - MQRC_MSG_MARKED_BROWSE_CO_OP
      - MQRC_TRUNCATED_MSG_ACCEPTED
    - Alle anderen Ursachencodes bedeuten, dass die Nachricht nicht konvertiert wurde.

    Die folgende Verarbeitung gilt für die integrierten Formate; sie gilt nicht für benutzerdefinierte Formate:

11. Mit Ausnahme der folgenden Formate:
    - MQFMT_ADMIN
    - MQFMT_COMMAND_1
    - MQFMT_COMMAND_2
    - MQFMT_EVENT
    - MQFMT_IMS_VAR_STRING
    - MQFMT_PCF
    - MQFMT_STRING

    kann keines der integrierten Formate aus oder in Zeichensätze konvertiert werden, die keine Einzelbytezeichen für die Zeichen enthalten, die in Warteschlangennamen gültig sind. Wenn versucht wird, eine solche Konvertierung auszuführen, wird die Nachricht unkonvertiert mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_SOURCE_CCSID_ERROR oder MQRC_TARGET_CCSID_ERROR zurückgegeben.

    Der Unicode-Zeichensatz UCS-2 ist ein Beispiel für einen Zeichensatz, der keine Singlebytezeichen für die Zeichen enthält, die in Warteschlangennamen gültig sind.

12. Wenn die Nachrichtendaten für ein integriertes Format abgeschnitten werden, werden Felder in der Nachricht, die Zeichenfolgenlängen oder Elementanzahlen oder Strukturanzahlen enthalten, *nicht* angepasst, um die Länge der tatsächlich an die Anwendung zurückgegebenen Daten anzugeben. Die für solche Felder in den Nachrichtendaten zurückgegebenen Werte sind die Werte, die *vor dem Abschneiden* für die Nachricht gelten.

    Stellen Sie beim Verarbeiten von Nachrichten wie einer abgeschnittenen MQFMT_ADMIN-Nachricht sicher, dass die Anwendung nicht versucht, auf Daten nach dem Ende der zurückgegebenen Daten zuzugreifen.

13. Wenn der Formatname MQFMT_DEAD_LETTER_HEADER lautet, beginnen die Nachrichtendaten mit einer MQDLH-Struktur, möglicherweise gefolgt von null oder mehr Bytes mit Anwendungsnachrichtendaten. Das Format, der Zeichensatz und die Codierung der Anwendungsnachrichtendaten werden durch die Felder *Format*, *CodedCharSetId* und *Encoding* in der MQDLH-Struktur am Anfang der Nachricht definiert. Da die MQDLH-Struktur und Anwendungsnachrichtendaten unterschiedliche Zeichensätze und Codierungen aufweisen können, erfordert die MQDLH-Struktur und/oder die Anwendungsnachrichtendaten möglicherweise eine Konvertierung.

    Der Warteschlangenmanager konvertiert bei Bedarf zuerst die MQDLH-Struktur. Wenn die Konvertierung erfolgreich ist oder die MQDLH-Struktur keine Konvertierung erfordert, prüft der Warteschlangenmanager die Felder *CodedCharSetId* und *Encoding* in der MQDLH-Struktur, um festzustellen, ob eine Konvertierung der Anwendungsnachrichtendaten erforderlich ist. Wenn eine Konvertierung *erforderlich* ist, ruft der Warteschlangenmanager den benutzerdefinierten Exit mit dem durch das Feld *Format* angegebenen Namen in der MQDLH-Struktur auf oder führt die Konvertierung selbst aus (wenn *Format* der Name eines integrierten Formats ist).

Wenn der MQGET-Aufruf den Beendigungscode MQCC_WARNING und einen der Ursachencodes zurückgibt, die angeben, dass die Konvertierung nicht erfolgreich war, trifft eine der folgenden Aussagen zu:

- Die MQDLH-Struktur konnte nicht konvertiert werden. In diesem Fall wurden die Anwendungsnachrichtendaten ebenfalls nicht konvertiert.
- Die MQDLH-Struktur wurde konvertiert, die Anwendungsnachrichtendaten jedoch nicht.

Die Anwendung kann die in den Feldern *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* und die in der MQDLH-Struktur zurückgegebenen Werte prüfen, um zu ermitteln, welche der oben genannten Aussagen zutrifft.

14. Wenn der Formatname MQFMT_XMIT_Q_HEADER lautet, beginnen die Nachrichtendaten mit einer MQXQH-Struktur, möglicherweise gefolgt von null oder mehr Bytes mit zusätzlichen Daten. Diese zusätzlichen Daten sind in der Regel die Anwendungsnachrichtendaten (die eine Länge von Null haben können), es können jedoch am Anfang der zusätzlichen Daten auch eine oder mehrere weitere MQ-Headerstrukturen vorhanden sein.

    Die MQXQH-Struktur muss im Zeichensatz und in der Codierung des Warteschlangenmanagers vorhanden sein. Das Format, der Zeichensatz und die Codierung der Daten, die auf die MQXQH-Struktur folgen, werden durch die Felder *Format*, *CodedCharSetId* und *Encoding* in der *innerhalb* des MQXQH enthaltenen MQMD-Struktur angegeben. Für jede nachfolgende vorhandene MQ-Headerstruktur beschreiben die Felder *Format*, *CodedCharSetId* und *Encoding* in der Struktur die Daten, die auf diese Struktur folgen. Bei diesen Daten handelt es sich entweder um eine weitere MQ-Headerstruktur oder um die Anwendungsnachrichtendaten.

    Wenn die Option MQGMO_CONVERT für eine MQFMT_XMIT_Q_HEADER-Nachricht angegeben ist, werden die Anwendungsnachrichtendaten und bestimmte der MQ-Headerstrukturen konvertiert, *die Daten in der MQXQH-Struktur werden jedoch nicht konvertiert*. Dabei gilt bei der Rückgabe vom MQGET-Aufruf Folgendes:

    - Die Werte der Felder *Format*, *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* beschreiben die Daten in der MQXQH-Struktur und *nicht* die Anwendungsnachrichtendaten. Daher sind die Werte *nicht* mit den Werten identisch, die von der Anwendung angegeben wurden, die den MQGET-Aufruf ausgegeben hat.

      Dies führt dazu, dass eine Anwendung, die wiederholt Nachrichten von einer Übertragungswarteschlange empfängt, für die die Option MQGMO_CONVERT angegeben ist, die Felder *CodedCharSetId* und *Encoding* im Parameter *MsgDesc* auf die Werte zurücksetzen muss, die für die Anwendungsnachrichtendaten erforderlich sind. Dies muss vor jedem MQGET-Aufruf erfolgen.
    - Die Werte der Felder *Format*, *CodedCharSetId* und *Encoding* in der letzten MQ-Headerstruktur beschreiben die Anwendungsnachrichtendaten. Wenn keine anderen MQ-Headerstrukturen vorhanden sind, werden die Anwendungsnachrichtendaten durch diese Felder in der MQMD-Struktur innerhalb der MQXQH-Struktur beschrieben. Wenn die Konvertierung erfolgreich ist, sind die Werte mit den Werten identisch, die im Parameter *MsgDesc* von der Anwendung angegeben werden, die den MQGET-Aufruf ausgegeben hat.

    Wenn die Nachricht eine Verteilerlistennachricht ist, wird die MQXQH-Struktur von einer MQDH-Struktur gefolgt (plus ihrer Arrays von MQOR- und MQPMR-Datensätzen), die wiederum unter Umständen von null oder mehr weiteren MQ-Headerstrukturen und null oder mehr Bytes mit Anwendungsnachrichtendaten gefolgt wird. Wie die MQXQH-Struktur muss die MQDH-Struktur im Zeichensatz und in der Codierung des Warteschlangenmanagers vorhanden sein. Sie wird nicht im MQGET-Aufruf konvertiert, auch wenn die Option MQGMO_CONVERT angegeben ist.

    Die oben beschriebene Verarbeitung der MQXQH- und MQDH-Struktur ist im Wesentlichen für die Verwendung durch Nachrichtenkanalagenten vorgesehen, wenn diese Nachrichten von Übertragungswarteschlangen abrufen.

Konvertierung von Berichtsnachrichten

Im Allgemeinen kann eine Berichtsnachricht je nach den Berichtsoptionen, die der Absender der ursprünglichen Nachricht angegeben hat, unterschiedliche Mengen an Anwendungsnachrichtendaten enthalten.

Ein Aktivitätsbericht kann jedoch Daten enthalten, ohne dass die Berichtsoption * _WITH_DATA in der Konstanten erwähnt.

Insbesondere kann eine Berichtsnachricht Folgendes enthalten:

1. Keine Anwendungsnachrichtendaten
2. Einige der Anwendungsnachrichtendaten aus der ursprünglichen Nachricht

   Dies ist der Fall, wenn der Absender der ursprünglichen Nachricht MQRO_*_WITH_DATA angibt und die Nachricht länger als 100 Bytes ist.
3. Alle Anwendungsnachrichtendaten aus der ursprünglichen Nachricht

   Dies ist der Fall, wenn der Absender der ursprünglichen Nachricht MQRO_*_WITH_FULL_DATA angibt oder wenn er MQRO_*_WITH_DATA angibt und die Nachricht maximal 100 Bytes lang ist.

Wenn der Warteschlangenmanager oder Nachrichtenkanalagent eine Berichtsnachricht generiert, kopiert er den Formatnamen aus der ursprünglichen Nachricht in das Feld *Format* in den Steuerinformationen in der Berichtsnachricht. Daher schließt der Formatname in der Berichtsnachricht unter Umständen eine Datenlänge ein, die von der tatsächlich in der Berichtsnachricht vorhandenen Länge abweicht (siehe Fall 1 und 2 oben).

Wenn die Option MQGMO_CONVERT angegeben ist, wenn die Berichtsnachricht abgerufen wird:

- Für den oben genannten Fall 1 wird der Datenkonvertierungsexit nicht aufgerufen (da die Berichtsnachricht keine Daten enthält).
- Für den oben genannten Fall 3 schließt der Formatname die Länge der Nachrichtendaten ordnungsgemäß ein.
- Für den oben genannten Fall 2 wird der Datenkonvertierungsexit aufgerufen, um eine Nachricht zu konvertieren, die *kürzer* ist als die im Formatnamen eingeschlossene Länge.

  Darüber hinaus ist der an den Exit übergebene Ursachencode in der Regel MQRC_NONE (d. h. der Ursachencode gibt nicht an, dass die Nachricht abgeschnitten wurde). Dies geschieht, da die Nachrichtendaten vom *Absender* der Berichtsnachricht und nicht vom Warteschlangenmanager des Empfängers als Reaktion auf den MQGET-Aufruf abgeschnitten wurden.

Aufgrund dieser Möglichkeiten darf der Datenkonvertierungsexit den Formatnamen *nicht* verwenden, um die Länge der übergebenen Daten abzuleiten. Stattdessen muss der Exit die Länge der bereitgestellten Daten prüfen und darauf vorbereitet sein, *weniger* Daten zu konvertieren als die durch den Formatnamen implizierte Länge. Wenn die Daten erfolgreich konvertiert werden können, müssen der Beendigungscode MQCC_OK und der Ursachencode MQRC_NONE vom Exit zurückgegeben werden. Die Länge der zu konvertierenden Nachrichtendaten wird als Parameter *InBufferLength* an den Exit übergeben.

Produktabhängige Programmierschnittstelle

MQDXP - Parameter des Datenkonvertierungsexits

Die MQDXP-Struktur ist ein Parameter, den der Warteschlangenmanager an den Datenkonvertierungsexit übergibt, wenn der Exit aufgerufen wird, um Daten im Rahmen der Verarbeitung des Aufrufs MQGET zu konvertieren. Weitere Informationen zum Datenkonvertierungsexit finden Sie in der Beschreibung des MQ_DATA_CONV_EXIT-Aufrufs.

Zeichendaten in MQDXP befinden sich im Zeichensatz des lokalen Warteschlangenmanagers, der durch das Warteschlangenmanagerattribut *CodedCharSetId* angegeben wird. Numerische Daten in MQDXP befinden sich in der nativen Maschinencodierung, die durch MQENC_NATIVE angegeben wird.

Nur die Felder *DataLength*, *CompCode*, *Reason* und *ExitResponse* in der MQDXP-Struktur können vom Exit geändert werden. Änderungen an anderen Feldern werden ignoriert. Das Feld *DataLength* kann jedoch *nicht* geändert werden, wenn die konvertierte Nachricht ein Segment ist, das nur einen Teil einer logischen Nachricht enthält.

Wenn die Steuerung vom Exit wieder an den Warteschlangenmanager übergeht, prüft der Warteschlangenmanager die zurückgegebenen Werte in MQDXP. Wenn die zurückgegebenen Werte nicht gültig sind,

setzt der Warteschlangenmanager die Verarbeitung so fort, als hätte der Exit MQXDR_CONVERSION_FAILED in *ExitResponse* zurückgegeben. Jedoch ignoriert der Warteschlangenmanager in diesem Fall die vom Exit zurückgegebenen Werte der Felder *CompCode* und *Reason* und verwendet stattdessen die Warte, die diese Felder bei der *Eingabe* in den Exit hatten. Die folgenden Werte in MQDXP führen dazu, dass diese Verarbeitung ausgeführt wird:

- Das Feld *ExitResponse* enthält nicht MQXDR_OK und nicht MQXDR_CONVERSION_FAILED
- Das Feld *CompCode* enthält nicht MQCC_OK und nicht MQCC_WARNING
- Das Feld *DataLength* ist kleiner als Null oder das Feld *DataLength* wird geändert, wenn die konvertierte Nachricht ein Segment ist, das nur einen Teil einer logischen Nachricht enthält.

Die folgende Tabelle enthält eine Zusammenfassung der Felder in der Struktur.

Tabelle 579. Felder in MQDXP

Feld	Beschreibung	Thema
StrucId	Struktur-ID	StrucId
Version	Strukturversionsnummer	Version
AppOptions	Anwendungsoptionen	AppOptions
Encoding	Numerische Codierung, die von der Anwendung angefordert wird	Codierung
CodedCharSetId	Zeichensatz, der von der Anwendung angefordert wird	CodedCharSetId
DataLength	Länge der Nachrichtendaten in Byte	DataLength
CompCode	Beendigungscode	CompCode
Reason	Ursachencode, der *CompCode* qualifiziert	Reason
ExitResponse	Antwort vom Exit	ExitResponse
Hconn	Verbindungskennung	Hconn
pEntryPoints	Adresse der MQIEP-Struktur	pEntryPoints

Felder

Die MQDXP-Struktur enthält die folgenden Felder, die in alphabetischer Reihenfolge beschrieben werden.

AppOptions
Typ: MQLONG

Dies ist eine Kopie des Feldes *Options* der MQGMO-Struktur, die von der Anwendung angegeben wird, die den MQGET-Aufruf ausgibt. Der Exit muss diese Werte möglicherweise prüfen, um festzustellen, ob die Option MQGMO_ACCEPT_TRUNCATED_MSG angegeben wurde.

Dies ist ein Eingabefeld für den Exit.

CodedCharSetId
Typ: MQLONG

Dies ist die codierte Zeichensatz-ID des Zeichensatzes, der von der Anwendung angefordert wird, die den MQGET-Aufruf ausgibt. Weitere Informationen finden Sie im Feld *CodedCharSetId* in der MQMD-Struktur. Wenn die Anwendung den Sonderwert MQCCSI_Q_MGR im MQGET-Aufruf angibt, ändert der Warteschlangenmanager diesen Wert in die tatsächliche Zeichensatz-ID des vom Warteschlangenmanagers verwendeten Zeichensatzes, bevor er den Exit aufruft.

Wenn die Konvertierung erfolgreich ist, muss der Exit diesen Wert in das Feld *CodedCharSetId* im Nachrichtendeskriptor kopieren.

Dies ist ein Eingabefeld für den Exit.

CompCode
Typ: MQLONG

Wenn der Exit aufgerufen wird, enthält dieses Feld den Beendigungscode, der an die Anwendung zurückgegeben wird, die den MQGET-Aufruf ausgegeben hat, wenn der Exit keine Aktionen ausführt. Der Wert ist immer MQCC_WARNING, da die Nachricht entweder abgeschnitten wurde oder eine Konvertierung erfordert, die noch nicht ausgeführt wurde.

Bei der Ausgabe vom Exit enthält dieses Feld den Beendigungscode, der an die Anwendung zurückgegeben werden soll, im Parameter *CompCode* des MQGET-Aufrufs. Nur MQCC_OK und MQCC_WARNING sind gültig. Vorschläge dazu, wie der Exit dieses Feld bei der Ausgabe festlegen kann, finden Sie in der Beschreibung des Felds *Reason*.

Dies ist ein Ein-/Ausgabefeld für den Exit.

DataLength
Typ: MQLONG

Wenn der Exit aufgerufen wird, enthält dieses Feld die ursprüngliche Länge der Anwendungsnachrichtendaten. Wenn die Nachricht abgeschnitten wurde, damit sie in den von der Anwendung bereitgestellten Puffer passt, ist die für den Exit bereitgestellte Nachricht *kleiner* als der Wert von *DataLength*. Die Größe der an den Exit bereitgestellten Nachricht wird unabhängig von ausgeführten Abschneidevorgängen immer durch den Parameter *InBufferLength* angegeben.

Das Abschneiden wird dadurch angegeben, dass das Feld *Reason* bei der Eingabe im Exit den Wert QRC_TRUNCATED_MSG_ACCEPTED aufweist.

Die meisten Konvertierungen müssen diese Länge nicht ändern. Ein Exit kann die Länge aber bei Bedarf ändern. Der vom Exit festgelegte Wert wird im Parameter *DataLength* des MQGET-Aufrufs an die Anwendung zurückgegeben. Diese Länge kann jedoch *nicht* geändert werden, wenn die konvertierte Nachricht ein Segment ist, das nur einen Teil einer logischen Nachricht enthält. Dies hängt damit zusammen, dass eine Änderung der Länge dazu führen würde, dass der Versatz nachfolgende Segmente in der logischen Nachricht falsch wäre.

Hinweis: Wenn der Exit die Länge der Daten ändern möchte, achten Sie darauf, dass der Warteschlangenmanager bereits entschieden hat, ob die Nachrichtendaten in den Anwendungspuffer passen. Diese Entscheidung erfolgt anhand der Länge der *unkonvertierten* Daten. Diese Entscheidung legt fest, ob die Nachricht aus der Warteschlange entfernt wird (oder bei einer Anzeigeanforderung der Anzeigecursor verschoben wird) und nicht von Änderungen der Datenlänge betroffen ist, die durch die Konvertierung verursacht werden. Aus diesem Grund wird empfohlen, dass Konvertierungsexits keine Änderung an der Länge der Anwendungsnachrichtendaten vornehmen.

Wenn die Zeichenkonvertierung eine Änderung der Länge beinhaltet, kann eine Zeichenfolge bei Bedarf mit derselben Länge in Bytes, durch Abschneiden abschließender Leerzeichen oder durch das Auffüllen mit Leerzeichen in eine andere Zeichenfolge konvertiert werden.

Der Exit wird nicht aufgerufen, wenn die Nachricht keine Anwendungsnachrichtendaten enthält. Daher ist *DataLength* immer größer als Null.

Dies ist ein Ein-/Ausgabefeld für den Exit.

Encoding
Typ: MQLONG

Numerische Codierung, die von der Anwendung angefordert wird.

Dies ist die numerische Codierung, die von der Anwendung angefordert wird, die den MQGET-Aufruf ausgibt. Weitere Informationen finden Sie im Feld *Encoding* in der MQMD-Struktur.

Wenn die Konvertierung erfolgreich ist, kopiert der Exit diesen Wert in das Feld *Encoding* im Nachrichtendeskriptor.

Dies ist ein Eingabefeld für den Exit.

ExitOptions
Typ: MQLONG

Dies ist ein reserviertes Feld. Der Wert lautet `0`.

ExitResponse
Typ: MQLONG

Antwort vom Exit.Dies wird vom Exit festgelegt, um den Erfolg oder das Fehlschlagen der Konvertierung anzugeben. Folgende Werte sind möglich:

MQXDR_OK
Die Konvertierung war erfolgreich.

Wenn der Exit diesen Wert angibt, gibt der Warteschlangenmanager Folgendes an die Anwendung zurück, die den MQGET-Aufruf ausgegeben hat:

- Den Wert des Felds *CompCode* bei der Ausgabe vom Exit
- Den Wert des Felds *Reason* bei der Ausgabe vom Exit
- Den Wert des Felds *DataLength* bei der Ausgabe vom Exit
- Den Inhalt des Ausgabepuffers des Exits, *OutBuffer*. Die Anzahl der zurückgegebenen Bytes ist der Parameter *OutBufferLength* oder der Wert des Feldes *DataLength* bei der Ausgabe vom Exit, je nachdem, welcher Wert niedriger ist.

  Wenn die Felder *Encoding* und *CodedCharSetId* im Nachrichtendeskriptorparameter des Exits *beide* unverändert sind, gibt der Warteschlangenmanager Folgendes zurück:
- Den Wert der Felder *Encoding* und *CodedCharSetId* in der MQDXP-Struktur bei der *Eingabe* in den Exit.

  Wenn eines oder beide der Felder *Encoding* und *CodedCharSetId* im Nachrichtendeskriptorparameter des Exits geändert wurden, gibt der Warteschlangenmanager Folgendes zurück:
- Den Wert der Felder *Encoding* und *CodedCharSetId* im Nachrichtendeskriptorparameter des Exits bei der Ausgabe vom Exit

MQXDR_CONVERSION_FAILED
Die Konvertierung war nicht erfolgreich.

Wenn der Exit diesen Wert angibt, gibt der Warteschlangenmanager Folgendes an die Anwendung zurück, die den MQGET-Aufruf ausgegeben hat:

- Den Wert des Felds *CompCode* bei der Ausgabe vom Exit
- Den Wert des Felds *Reason* bei der Ausgabe vom Exit
- Den Wert des Felds *DataLength* bei der *Eingabe* in den Exit
- Den Inhalt des Eingabepuffers des Exits, *InBuffer*. Die Anzahl der zurückgegebenen Bytes wird durch den Parameter *InBufferLength* angeben

Wenn der Exit *InBuffer* geändert hat, sind die Ergebnisse nicht definiert.

ExitResponse ist ein Ausgabefeld für den Exit.

Hconn
Type: MQHCONN

Dies ist eine Verbindungskennung, die im MQXCNVC-Aufruf verwendet werden kann. Diese Kennung ist nicht unbedingt mit der Kennung identisch, die von der Anwendung angegeben wurde, die den MQGET-Aufruf ausgegeben hat.

pEntryPoints
Type: PMQIEP

Die Adresse einer MQIEP-Struktur, über die MQI- und DCI-Aufrufe möglich sind.

Reason
Typ: MQLONG

Ursachencode zur Qualifikation von *CompCode*.

Wenn der Exit aufgerufen wird, enthält dieses Feld den Ursachencode, der an die Anwendung zurückgegeben wird, die den MQGET-Aufruf ausgegeben hat, wenn der Exit keine Aktionen ausführt. Zu den möglichen Werten gehört MQRC_TRUNCATED_MSG_ACCEPTED. Dieser Wert gibt an, dass die Nachricht abgeschnitten wurde, damit sie in den von der Anwendung bereitgestellten Puffer passt. Ein weiterer möglicher Wert ist MQRC_NOT_CONVERTED. Dieser Wert gibt an, dass die Nachricht eine Konvertierung erfordert, die noch nicht ausgeführt wurde.

Bei der Ausgabe vom Exit enthält dieses Feld den Ursachencode, der an die Anwendung zurückgegeben werde soll, im Parameter *Reason* des MQGET-Aufrufs. Folgendes wird empfohlen:

- Wenn *Reason* bei der Eingabe in den Exit den Wert MQRC_TRUNCATED_MSG_ACCEPTED hatte, dürfen die Felder *Reason* und *CompCode* unabhängig davon, ob die Konvertierung erfolgreich ist oder fehlschlägt, nicht geändert werden.

  (Wenn das Feld *CompCode* nicht den Wert MQCC_OK enthält, kann die Anwendung, die die Nachricht abruft, einen Konvertierungsfehler ermitteln, indem sie die zurückgegebenen Werte für *Encoding* und *CodedCharSetId* im Nachrichtendeskriptor mit den angeforderten Werten vergleicht. Im Gegensatz dazu kann die Anwendung eine abgeschnittene Nachricht nicht von einer Nachricht unterscheiden, die in den Puffer gepasst hat. Aus diesem Grund ist MQRC_TRUNCATED_MSG_ACCEPTED vorrangig zu allen anderen Ursachen zurückzugeben, die einen Konvertierungsfehler angeben.)
- Wenn *Reason* bei der Eingabe in den Exit einen anderen Wert hatte:
  - Wenn die Konvertierung erfolgreich ist, muss *CompCode* auf MQCC_OK und *Reason* auf MQRC_NONE festgelegt werden.
  - Wenn die Konvertierung fehlschlägt oder die Nachricht länger wird und abgeschnitten werden muss, damit sie in den Puffer passt, muss *CompCode* auf MQCC_WARNING festgelegt werden (oder unverändert bleiben) und *Reason* muss auf einen der aufgeführten Werte festgelegt werden, um die Art des Fehlers anzugeben.

    Hinweis: Wenn die Nachricht nach der Konvertierung zu groß für den Puffer ist, muss sie nur abgeschnitten werden, wenn die Anwendung, die den MQGET-Aufruf ausgegeben hat, die Option MQGMO_ACCEPT_TRUNCATED_MSG angegeben hat:

    - Wurde diese Option nicht angegeben, wird die Ursache MQRC_TRUNCATED_MSG_ACCEPTED zurückgegeben.
    - Wenn die Anwendung diese Option nicht angegeben hat, wird die Nachricht unkonvertiert mit dem Ursachencode MQRC_CONVERTED_MSG_TOO_BIG zurückgegeben.

Die aufgeführten Ursachencodes werden für die Verwendung durch den Exit empfohlen, um den Grund für das Fehlschlagen der Konvertierung anzugeben. Der Exit kann jedoch andere Werte aus der Gruppe der MQRC_*-Codes zurückgegeben, wenn dies als angemessen erachtet wird. Darüber hinaus ist der Wertebereich von MQRC_APPL_FIRST bis MQRC_APPL_LAST für die Verwendung durch den Exit zugewiesen, um Bedingungen anzugeben, die der Exit der Anwendung mitteilen möchte, die den MQGET-Aufruf ausgibt.

Anmerkung: Wenn die Nachricht nicht erfolgreich konvertiert werden kann, *muss* der Exit MQXDR_CONVERSION_FAILED im Feld *ExitResponse* zurückgeben, damit der Warteschlangenmanager die unkonvertierte Nachricht zurückgibt. Dies gilt unabhängig von dem Ursachencode, der im Feld *Reason* zurückgegeben wird.

MQRC_APPL_FIRST
(900, X'384') Niedrigster Wert für den anwendungsdefinierten Ursachencode.

MQRC_APPL_LAST
(999, X'3E7') Höchster Wert für den anwendungsdefinierten Ursachencode.

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Die konvertierten Daten sind zu groß für den Puffer.

MQRC_NOT_CONVERTED
(2119, X'847') Die Nachrichtendaten wurden nicht konvertiert.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Die quellcodierte Zeichensatz-ID ist nicht gültig.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codierung gepackter Dezimalzahlen in der Nachricht wurde nicht erkannt.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codierung von Gleitkommazahlen in der Nachricht wurde nicht erkannt.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Quellcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') Die zielcodierte Zeichensatz-ID ist nicht gültig.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Durch Empfänger angegebene Ganzzahlcodierung nicht erkannt.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Durch Empfänger angegebene Gleitkommacodierung nicht erkannt.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Zielcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Abgeschnittene Nachricht zurückgegeben (Verarbeitung ist abgeschlossen).

Dies ist ein Ein-/Ausgabefeld für den Exit.

StrucId
Type: MQCHAR4

Struktur-ID.Folgende Werte sind möglich:

MQDXP_STRUC_ID
ID für die Parameterstruktur des Datenkonvertierungsexits.

Für die Programmiersprache C ist auch die Konstante MQDXP_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQDXP_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Eingabefeld für den Exit.

Version
Typ: MQLONG

Strukturversionsnummer.Folgende Werte sind möglich:

MQDXP_VERSION_1
Versionsnummer für die Parameterstruktur des Datenkonvertierungsexits.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQDXP_CURRENT_VERSION
Aktuelle Version der Parameterstruktur des Datenkonvertierungsexits.

Anmerkung: Wenn eine neue Version dieser Struktur eingeführt wird, wird das Layout des vorhandenen Teils nicht geändert. Daher muss der Exit prüfen, ob das Feld *Version* größer oder gleich der niedrigsten Version ist, die die Felder enthält, die der Exit verwenden muss.

Dies ist ein Eingabefeld für den Exit.

Deklaration in Programmiersprache 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 */
};
```

COBOL-Deklaration (nur 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.
```

Deklaration in System/390 Assembler

```
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 – Zeichen konvertieren

Der MQXCNVC-Aufruf konvertiert Zeichen mit der Programmiersprache C aus einem Zeichensatz in einen anderen.

Dieser Aufruf ist Bestandteil der WebSphere MQ Data Conversion Interface (DCI). Dabei handelt es sich um eine der WebSphere MQ-Frameworkschnittstellen.

Hinweis: Der Aufruf kann aus Anwendungsumgebungen sowie aus Umgebungen mit Datenkonvertierungsexits verwendet werden.

Syntax

MQXCNVC (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parameter

Hconn
Typ: MQHCONN - Eingabe

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager.

In einem Datenkonvertierungsexit ist *Hconn* normalerweise die Kennung, die im Feld *Hconn* der MQDXP-Struktur an den Datenkonvertierungsexit übergeben wird. Diese Kennung ist nicht unbedingt mit der Kennung identisch, die von der Anwendung angegeben wurde, die den MQGET-Aufruf ausgegeben hat.

Unter IBM ikann der folgende Sonderwert für *Hconn*angegeben werden:

MQHC_DEF_HCONN
Standardverbindungskennung

Wenn Sie eine CICS TS 3.2-Anwendung oder höher ausführen, stellen Sie sicher, dass das Exitprogramm für die Zeichenkonvertierung, das den Aufruf MQXCNVC aufruft, als OPENAPI definiert ist. Diese Definition verhindert, dass der Fehler 2018 MQRC_HCONN_ERROR auftritt, der durch eine falsche Verbindung verursacht wird, und ermöglicht den Abschluss des Aufrufs MQGET.

Optionen
Typ: MQLONG - Eingabe

Optionen zur Steuerung der Aktion von MQXCNVC.

Es können keine oder mehrere der beschriebenen Optionen angegeben werden. Werden mehrere Optionen angegeben, können die Werte:

- Hinzufügen (fügen Sie dieselbe Konstante nicht mehrmals hinzu) oder
- Mit der bitweisen ODER-Operation kombiniert werden (wenn die Programmiersprache Bitoperationen unterstützt)

Standardkonvertierungsoption: Die folgende Option steuert die Verwendung der Standardzeichenkonvertierung:

MQDCC_DEFAULT_CONVERSION
Standardkonvertierung.

Diese Option gibt an, dass die Standardzeichenkonvertierung verwendet werden kann, wenn einer oder beide der im Aufruf angegebenen Zeichensätze nicht unterstützt werden. Dadurch kann der Warteschlangenmanager einen installationsspezifischen Standardzeichensatz verwenden, der sich bei der Konvertierung der Zeichenfolge an den angegebenen Zeichensatz annähert.

Anmerkung: Die Verwendung eines angenäherten Zeichensatzes zur Konvertierung der Zeichenfolge hat zur Folge, dass einige Zeichen möglicherweise nicht richtig konvertiert werden. Dies kann verhindert werden, indem in der Zeichenfolge nur Zeichen verwendet werden, die sowohl im angegebenen Zeichensatz als auch im Standardzeichensatz vorkommen.

Die Standardzeichensätze werden durch eine Konfigurationsoption definiert, wenn der Warteschlangenmanager installiert oder erneut gestartet wird.

Wenn MQDCC_DEFAULT_CONVERSION nicht angegeben ist, verwendet der Warteschlangenmanager nur die angegebenen Zeichensätze zum Konvertieren der Zeichenfolge und der Aufruf schlägt fehl, wenn einer oder beide der Zeichensätze nicht unterstützt werden.

Diese Option wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Auffülloption: Mit der folgenden Option kann der Warteschlangenmanager die konvertierte Zeichenfolge mit Leerzeichen auffüllen oder nicht relevante abschließende Leerzeichen löschen, damit die konvertierte Zeichenfolge in den Zielpuffer passt:

MQDCC_FILL_TARGET_BUFFER
Zielpuffer auffüllen.

Für diese Option ist es erforderlich, dass die Konvertierung so ausgeführt wird, dass der Zielpuffer vollständig gefüllt ist:

- Wenn die Zeichenfolge beim Konvertieren kürzer wird, werden Leerzeichen hinzugefügt, um den Zielpuffer zu füllen.
- Wenn die Zeichenfolge beim Konvertieren länger wird, werden abschließende Zeichen, die nicht relevant sind, gelöscht, damit die konvertierte Zeichenfolge in den Zielpuffer passt. Wenn dies erfolgreich ausgeführt werden kann, wird der Aufruf mit MQCC_OK und dem Ursachencode MQRC_NONE abgeschlossen.

  Wenn zu wenige nicht relevante abschließende Zeichen vorhanden sind, wird so viel wie möglich der Zeichenfolge im Zielpuffer abgelegt und der Aufruf wird mit MQCC_WARNING und dem Ursachencode MQRC_CONVERTED_MSG_TOO_BIG abgeschlossen.

  Nicht relevante Zeichen sind:

  – Abschließende Leerzeichen
  – Zeichen, die auf das erste Nullzeichen in der Zeichenfolge folgen (jedoch ausgenommen des ersten Nullzeichens selbst)
- Wenn die Zeichenfolge, *TargetCCSID* und *TargetLength* so aussehen, dass der Zielpuffer nicht vollständig mit gültigen Zeichen gefüllt werden kann, schlägt der Aufruf mit MQCC_FAILED und dem Ursachencode MQRC_TARGET_LENGTH_ERROR fehl. Dies kann der Fall sein, wenn *TargetCCSID* ein reiner DBCS-Zeichensatz ist (z. B. UCS-2), *TargetLength* jedoch eine Länge angibt, die aus einer ungeraden Zahl an Bytes besteht.
- *TargetLength* kann kleiner oder größer als *SourceLength* sein. Bei der Rückgabe von MQXCNVC hat *DataLength* den gleichen Wert wie *TargetLength*.

Wenn diese Option nicht angegeben ist, gilt Folgendes:

- Die Zeichenfolge kann innerhalb des Zielpuffers nach Bedarf kürzer oder länger werden. Nicht relevante abschließende Zeichen werden nicht hinzugefügt oder gelöscht.

  Wenn die konvertierte Zeichenfolge in den Zielpuffer passt, wird der Aufruf mit MQCC_OK und dem Ursachencode MQRC_NONE abgeschlossen.

  Wenn die konvertierte Zeichenfolge zu groß für den Zielpuffer ist, wird so viel wie möglich von der Zeichenfolge im Zielpuffer abgelegt und der Aufruf wird mit MQCC_WARNING und dem Ursachencode MQRC_CONVERTED_MSG_TOO_BIG abgeschlossen. Beachten Sie, dass in diesem Fall weniger Bytes als *TargetLength* zurückgegeben werden können.
- *TargetLength* kann kleiner oder größer als *SourceLength* sein. Bei der Rückgabe von MQXCNVC ist *DataLength* kleiner oder gleich *TargetLength*.

Diese Option wird in den folgenden Umgebungen unterstützt: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Codierungsoptionen: Die beschriebenen Optionen können verwendet werden, um die Ganzzahlcodierungen der Quell- und Zielzeichenfolgen anzugeben. Die relevante Codierung wird *nur* verwendet, wenn die entsprechende Zeichensatz-ID angibt, dass die Darstellung des Zeichensatzes im Hauptspeicher von der für binäre ganze Zahlen verwendeten Codierung abhängt. Dies gilt nur für bestimmte Mehrbytezeichensätze (z. B. UCS-2-Zeichensätze).

Die Codierung wird ignoriert, wenn der Zeichensatz ein Einzelbytezeichensatz (SBCS) oder ein Mehrbytezeichensatz mit einer Darstellung im Hauptspeicher ist, die nicht von der Ganzzahlcodierung abhängt.

Nur einer der MQDCC_SOURCE_*-Werte muss angegeben werden, kombiniert mit einem der MQDCC_TARGET_*-Werte:

MQDCC_SOURCE_ENC_NATIVE
Die Quellcodierung ist der Standard für die Umgebung und Programmiersprache.

MQDCC_SOURCE_ENC_NORMAL
Die Quellcodierung ist normal.

MQDCC_SOURCE_ENC_REVERSED
Die Quellcodierung ist umgekehrt.

MQDCC_SOURCE_ENC_UNDEFINED
Die Quellcodierung ist nicht definiert.

MQDCC_TARGET_ENC_NATIVE
Die Zielcodierung ist der Standard für die Umgebung und Programmiersprache.

MQDCC_TARGET_ENC_NORMAL
Die Zielcodierung ist normal.

MQDCC_TARGET_ENC_REVERSED
Die Zielcodierung ist umgekehrt.

MQDCC_TARGET_ENC_UNDEFINED
Die Zielcodierung ist nicht definiert.

Die zuvor definierten Codierungswerte können direkt zum Feld *Options* hinzugefügt werden. Wenn die Quell- oder Zielcodierung jedoch aus dem Feld *Encoding* in der MQMD-Struktur oder einer anderen Struktur abgerufen wird, muss die folgende Verarbeitung ausgeführt werden:

1. Die Ganzzahlcodierung muss aus dem Feld *Encoding* extrahiert werden, indem die Codierungen von Gleitkommazahlen und gepackten Dezimalzahlen entfernt werden. Weitere Informationen hierzu finden Sie unter „Codierungen analysieren” auf Seite 903.
2. Die aus Schritt 1 resultierende Ganzzahlcodierung muss mit dem entsprechenden Faktor multipliziert werden, bevor sie zum Feld *Options* hinzugefügt wird. Diese Faktoren sind:
   - MQDCC_SOURCE_ENC_FACTOR für die Quellcodierung
   - MQDCC_TARGET_ENC_FACTOR für die Zielcodierung

Im folgenden Beispielcode ist dargestellt, wie dies in der Programmiersprache C codiert werden könnte:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_SOURCE_ENC_FACTOR
         + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_TARGET_ENC_FACTOR;
```

Sofern nicht anderweitig angegeben, sind die Codierungsoptionen standardmäßig nicht definiert (MQDCC_*_ENC_UNDEFINED). In den meisten Fällen wirkt sich dies nicht auf den erfolgreichen Abschluss des MQXCNVC-Aufrufs aus. Wenn der entsprechende Zeichensatz jedoch ein Mehrbytezeichensatz mit einer Darstellung ist, die von der Codierung abhängt (z. B. ein UCS-2-Zeichensatz), schlägt der Aufruf mit dem Ursachencode MQRC_SOURCE_INTEGER_ENC_ERROR oder MQRC_TARGET_INTEGER_ENC_ERROR fehl.

Die Codierungsoptionen werden in den folgenden Umgebungen unterstützt: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows.

Standardoption: Wenn keine der zuvor beschriebenen Optionen angegeben ist, kann die folgende Option verwendet werden:

MQDCC_NONE
Keine Optionen angegeben.

MQDCC_NONE ist zur Unterstützung der Programmdokumentation definiert. Diese Option ist nicht zur Verwendung mit einer anderen Option gedacht; da sie jedoch den Wert null hat, kann eine solche Verwendung nicht erkannt werden.

SourceCCSID
Typ: MQLONG - Eingabe

Dies ist die codierte Zeichensatz-ID der Eingabezeichenfolge in *SourceBuffer*.

SourceLength
Typ: MQLONG - Eingabe

Dies ist die Länge der Eingabezeichenfolge in *SourceBuffer* in Bytes; der Wert muss Null oder größer sein.

SourceBuffer
Typ: MQCHARxSourceLength - Eingabe

Dies ist der Puffer, der die Zeichenfolge enthält, die aus einem Zeichensatz in einen anderen konvertiert werden soll.

TargetCCSID
Typ: MQLONG - Eingabe

Dies ist die codierte Zeichensatz-ID des Zeichensatzes, in den *SourceBuffer* konvertiert werden soll.

TargetLength
Typ: MQLONG - Eingabe

Dies ist die Länge des Ausgabepuffers *TargetBuffer* in Bytes; der Wert muss Null oder größer sein. Der Wert kann kleiner oder größer als *SourceLength* sein.

TargetBuffer
Typ: MQCHARxTargetLength - Ausgabe

Dies ist die Zeichenfolge nach der Konvertierung in den durch *TargetCCSID* definierten Zeichensatz. Die konvertierte Zeichenfolge kann kürzer oder länger als die unkonvertierte Zeichenfolge sein. Der Parameter *DataLength* gibt die Anzahl gültiger zurückgegebener Bytes an.

DataLength
Typ: MQLONG - Ausgabe

Dies ist die Länge der im Ausgabepuffer *TargetBuffer* zurückgegebenen Zeichenfolge. Die konvertierte Zeichenfolge kann kürzer oder länger als die unkonvertierte Zeichenfolge sein.

CompCode
Typ: MQLONG - Ausgabe

Folgende Werte sind möglich:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode zur Qualifikation von *CompCode*.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Die konvertierten Daten sind zu groß für den Puffer.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Parameter Datenlänge ungültig.

MQRC_DBCS_ERROR
(2150, X'866') DBCS-Zeichenfolge ungültig.

MQRC_HCONN_ERROR
(2018, X'7E2') Verbindungskennung ungültig

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Nicht genug Systemressourcen verfügbar

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') Quellenpufferparameter ungültig.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Die quellcodierte Zeichensatz-ID ist nicht gültig.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Quellcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Quellenlängenparameter ungültig.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Nicht genug Speicher verfügbar

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Zielpufferparameter ungültig.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') Die zielcodierte Zeichensatz-ID ist nicht gültig.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Zielcodierung von Ganzzahlen wurde nicht erkannt.

MQRC_TARGET_LENGTH_ERROR
(2144, X'860') Ziellängenparameter ungültig.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unerwarteter Fehler aufgetreten

Ausführliche Informationen zu diesen Codes finden Sie unter Ursachencodes.

C-Aufruf

```
MQXCNVC (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
        TargetCCSID, TargetLength, TargetBuffer, &DataLength,
        &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 */
```

COBOL-Deklaration (nur IBM i)

```
     CALL 'MQXCNVC' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                          SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                          TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Deklarieren Sie die Parameter wie folgt:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Options that control the action of MQXCNVC
 01  OPTIONS       PIC S9(9) BINARY.
**   Coded character set identifier of string before conversion
 01  SOURCECCSID   PIC S9(9) BINARY.
**   Length of string before conversion
 01  SOURCELENGTH  PIC S9(9) BINARY.
**   String to be converted
 01  SOURCEBUFFER  PIC X(n).
**   Coded character set identifier of string after conversion
 01  TARGETCCSID   PIC S9(9) BINARY.
**   Length of output buffer
 01  TARGETLENGTH  PIC S9(9) BINARY.
**   String after conversion
 01  TARGETBUFFER  PIC X(n).
**   Length of output string
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

S/390-Assemblerdeklaration

```
         CALL MQXCNVC,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,         X
               SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER,     X
               DATALENGTH,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Datenkonvertierungsexit

Der MQ_DATA_CONV_EXIT-Aufruf beschreibt die Parameter, die an den Datenkonvertierungsexit übergeben werden.

Vom Warteschlangenmanager wird kein Einstiegspunkt mit dem Namen MQ_DATA_CONV_EXIT bereitgestellt (siehe Verwendungshinweis 11).

Diese Definition ist Bestandteil der WebSphere MQ Data Conversion Interface (DCI). Dabei handelt es sich um eine der WebSphere MQ-Frameworkschnittstellen.

Syntax

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parameter

DataConvExitParms
Typ: MQDXP - Ein-/Ausgabe

Diese Struktur enthält Informationen zum Aufruf des Exits. Der Exit legt Informationen in dieser Struktur fest, um das Ergebnis der Konvertierung anzugeben. Ausführliche Informationen zu den Feldern in dieser Struktur finden Sie unter „MQDXP - Parameter des Datenkonvertierungsexits" auf Seite 915.

MsgDesc
Typ: MQMD - Ein-/Ausgabe

Bei der Ausgabe an den Exit ist dies der Nachrichtendeskriptor, der den Nachrichtendaten zugeordnet ist, die im Parameter *InBuffer* an den Exit übergeben werden.

Anmerkung: Der Parameter *MsgDesc*, der an den Exit übergeben wird, ist immer die neueste MQMD-Version, die von dem Warteschlangenmanager unterstützt wird, der den Exit aufruft. Wenn der Exit zwischen verschiedenen Umgebungen portierbar sein soll, prüft der Exit das Feld *Version* in *MsgDesc*, um zu überprüfen, dass die Felder, die der Exit für den Zugriff benötigt, in der Struktur vorhanden sind.

In den folgenden Umgebungen wird an den Exit ein MQMD version-2 übergeben: AIX, HP-UX, IBM i, Solaris, Linux, Windows. In allen anderen Umgebungen, die den Datenkonvertierungsexit unterstützen, wird dem Exit ein MQMD der Version 1 übergeben.

Bei der Ausgabe ändert der Exit die Felder *Encoding* und *CodedCharSetId* in die von der Anwendung angeforderten Werte. Wenn die Konvertierung erfolgreich war, werden diese Änderungen an die Anwendung zurückgemeldet. Alle anderen Änderungen, die der Exit an der Struktur vornimmt, werden ignoriert und nicht an die Anwendung zurückgemeldet.

Wenn der Exit MQXDR_OK im Feld *ExitResponse* der MQDXP-Struktur zurückgibt, die Felder *Encoding* oder *CodedCharSetId* im Nachrichtendeskriptor jedoch nicht ändert, gibt der Warteschlangenmanager für diese Felder die Werte zurück, die die entsprechenden Felder in der MQDXP-Struktur bei der Eingabe in den Exit hatten.

InBufferLength
Typ: MQLONG - Eingabe

Länge von *InBuffer* in Bytes.

Dies ist die Länge des Eingabepuffers *InBuffer* und gibt die Anzahl der Bytes an, die vom Exit verarbeitet werden sollen. *InBufferLength* ist die Länge der Nachrichtendaten vor der Konvertierung oder die Länge des von der Anwendung im MQGET-Aufruf bereitgestellten Puffers, je nachdem, welcher Wert niedriger ist.

Der Wert ist immer größer als Null.

InBuffer
Typ: MQBYTExInBufferLength - Eingabe

Puffer, der die unkonvertierte Nachricht enthält.

Dieser Parameter enthält die Nachrichtendaten vor der Konvertierung. Wenn der Exit die Daten nicht konvertieren kann, gibt der Warteschlangenmanager die Inhalte dieses Puffers an die Anwendung zurück, wenn der Exit abgeschlossen wurde.

Anmerkung: Der Exit sollte *InBuffer* nicht ändern. Wenn dieser Parameter geändert wird, sind die Ergebnisse undefiniert.

In der Programmiersprache C ist dieser Parameter als untypisierter Zeiger definiert.

OutBufferLength
Typ: MQLONG - Eingabe

Länge von *OutBuffer* in Bytes.

Dies ist die Länge des Ausgabepuffers *OutBuffer*. Dieser Wert ist mit der Länge des von der Anwendung im MQGET-Aufruf bereitgestellten Puffers identisch.

Der Wert ist immer größer als Null.

OutBuffer
Typ: MQBYTExOutBufferLength - Ausgabe

Puffer, der die konvertierte Nachricht enthält.

Wenn die Konvertierung erfolgreich war (wie durch den Wert MQXDR_OK im Feld *ExitResponse* des Parameters *DataConvExitParms* angegeben, enthält *OutBuffer* bei der Ausgabe vom Exit die an die Anwendung zu übergebenden Daten in der angeforderten Darstellung. Wenn die Konvertierung nicht erfolgreich war, werden alle Änderungen, die der Exit an diesem Puffer vorgenommen hat, ignoriert.

In der Programmiersprache C ist dieser Parameter als untypisierter Zeiger definiert.

Hinweise zur Verwendung

1. Ein Datenkonvertierungsexit ist ein benutzerdefinierter Exit, der die Steuerung während der Verarbeitung eines MQGET-Aufrufs erhält. ´Die vom Datenkonvertierungsexit ausgeführte Funktion wird vom Bereitsteller des Exits definiert. Der Exit muss jedoch die hier und im MQDXP der zugehörigen Parameterstruktur beschriebenen Regeln einhalten.

   Welche Programmiersprachen für einen Datenkonvertierungsexit verwendet werden können, wird von der Umgebung festgelegt.
2. Der Exit wird nur aufgerufen, wenn *alle* der folgenden Bedingungen erfüllt sind:
   - Die Option MQGMO_CONVERT ist im MQGET-Aufruf angegeben
   - Das Feld *Format* im Nachrichtendeskriptor enthält nicht den Wert MQFMT_NONE
   - Die Nachricht liegt noch nicht in der erforderlichen Darstellung vor, d. h. einer oder beide Werte für *CodedCharSetId* und *Encoding* der Nachricht weichen von dem von der Anwendung in dem Nachrichtendeskriptor angegebenen Wert ab, der im MQGET-Aufruf bereitgestellt wird
   - Der Warteschlangenmanager hat die Konvertierung noch nicht erfolgreich ausgeführt.
   - Die Länge des Anwendungspuffers ist größer als Null
   - Die Länge der Nachrichtendaten ist größer als Null
   - Der Ursachencode während der MQGET-Operation ist MQRC_NONE oder MQRC_TRUNCATED_MSG_ACCEPTED
3. Wenn ein Exit geschrieben wird, sollten Sie überlegen, den Exit so zu codieren, dass er Nachrichten konvertieren kann, die abgeschnitten wurden. Abgeschnittene Nachrichten können wie folgt auftreten:
   - Die empfangende Anwendung stellt einen Puffer bereit, der kleiner ist als die Nachricht, gibt aber die Option MQGMO_ACCEPT_TRUNCATED_MSG im MQGET-Aufruf an.

     In diesem Fall hat das Feld *Reason* im Parameter *DataConvExitParms* bei der Ausgabe an den Exit den Wert MQRC_TRUNCATED_MSG_ACCEPTED.
   - Der Absender der Nachricht hat sie vor dem Senden abgeschnitten. Dies kann beispielsweise bei Berichtsnachrichten der Fall sein (weitere Informationen finden Sie unter „Konvertierung von Berichtsnachrichten” auf Seite 914).

In diesem Fall hat das Feld *Reason* im Parameter *DataConvExitParms* bei der Eingabe in den Exit den Wert MQRC_NONE (wenn die empfangende Anwendung einen Puffer bereitgestellt hat, der groß genug für die Nachricht war).

Daher kann der Wert des Feldes *Reason* bei der Eingabe in den Exit nicht immer herangezogen werden, um zu entscheiden, ob die Nachricht abgeschnitten wurde.

Das Unterscheidungsmerkmal einer abgeschnittenen Nachricht besteht darin, dass die Länge, die dem Exit im Parameter *InBufferLength* bereitgestellt wird, *kleiner* als die Länge ist, die durch den Formatnamen impliziert wird, der im Feld *Format* im Nachrichtendeskriptor enthalten ist. Daher sollte der Exit den Wert von *InBufferLength* prüfen, bevor er versucht, Daten zu konvertieren. Der Exit *sollte nicht* voraussetzen, dass alle durch den Formatnamen eingeschlossenen Daten bereitgestellt wurden.

Wenn der Exit *nicht* geschrieben wurde, um abgeschnittene Nachrichten zu codieren und *InBufferLength* kleiner ist als der erwartete Wert, gibt der Exit MQXDR_CONVERSION_FAILED im Feld *ExitResponse* des Parameters *DataConvExitParms* zurück, wobei die Felder *CompCode* und *Reason* auf MQCC_WARNING bzw. MQRC_FORMAT_ERROR festgelegt sind.

Wenn der Exit *geschrieben* wurde, um abgeschnittene Nachrichten zu konvertieren, konvertiert der Exit so viele Daten wie möglich (siehe nächster Verwendungshinweis), wobei er darauf achtet, nicht zu versuchen, Daten zu prüfen oder zu konvertieren, die über das Ende von *InBuffer* hinausgehen. Wenn die Konvertierung erfolgreich ausgeführt wird, nimmt der Exit keine Änderungen am Feld *Reason* im Parameter *DataConvExitParms* vor. Dies gibt MQRC_TRUNCATED_MSG_ACCEPTED zurück, wenn die Nachricht vom Warteschlangenmanager des Empfängers abgeschnitten wurde, und MQRC_NONE, wenn die Nachricht vom Absender der Nachricht abgeschnitten wurde.

Es ist auch möglich, dass eine Nachricht *während* der Konvertierung so viel länger wird, dass sie größer ist als *OutBuffer*. In diesem Fall muss der Exit entscheiden, ob die Nachricht abgeschnitten werden soll. Das Feld *AppOptions* im Parameter *DataConvExitParms* gibt an, ob die empfangende Anwendung die Option MQGMO_ACCEPT_TRUNCATED_MSG angegeben hat.

4. Im Allgemeinen werden alle oder keine der Daten, die dem Exit in *InBuffer* bereitgestellt werden, konvertiert. Eine Ausnahme von dieser Regel tritt jedoch ein, wenn die Nachricht vor oder während der Konvertierung abgeschnitten wird. In diesem Fall kann am Ende des Puffers ein unvollständiges Element vorhanden sein (z. B. 1 Byte eines Doppelbytezeichens oder 3 Byte einer 4-Byte-Ganzzahl. In diesem Fall sollten Sie überlegen, ob Sie das unvollständige Element auslassen und die nicht verwendeten Bytes in *OutBuffer* auf Nullen setzen. Vollständige Zeichen oder Zeichen innerhalb eines Bereichs oder einer Zeichenfolge *sollten* jedoch konvertiert werden.
5. Wenn ein Exit zum ersten Mal benötigt wird, versucht der Warteschlangenmanager, ein Objekt zu laden, das denselben Namen hat wie das Format (abgesehen von Erweiterungen). Das geladene Objekt muss den Exit enthalten, der Nachrichten mit diesem Formatnamen verarbeitet. Überlegen Sie, ob Sie den Exitnamen und den Namen des Objekts, das den Exit enthält, auf denselben Wert festlegen. Dies ist allerdings nicht in allen Umgebungen erforderlich.
6. Eine neue Kopie des Exits wird geladen, wenn eine Anwendung versucht, die erste Nachricht abzurufen, die dieses *Format* verwendet, seit die Anwendung eine Verbindung zum Warteschlangenmanager hergestellt hat. Für CICS- oder IMS-Anwendungen ist dies der Zeitpunkt, zu dem das CICS- oder IMS-Subsystem eine Verbindung zum Warteschlangenmanager hergestellt hat. Eine neue Kopie kann auch zu anderen Zeitpunkten geladen werden, wenn der Warteschlangenmanager eine zuvor geladene Kopie gelöscht hat. Aus diesem Grund darf ein Exit nicht versuchen, statischen Speicher für die Übertragung von Informationen von einem Aufruf des Exits an den nächsten zu verwenden – der Exit kann zwischen den beiden Aufrufen entladen werden.
7. Wenn ein benutzerdefinierter Exit mit demselben Namen wie eines der integrierten Formate vorhanden ist, die vom Warteschlangenmanager unterstützt werden, ersetzt der benutzerdefinierte Exit nicht die integrierte Konvertierungsroutine. Ein solcher Exit wird nur unter folgenden Umständen aufgerufen:

   - Wenn die integrierte Konvertierungsroutine Konvertierungen in oder aus den entsprechenden Werten für *CodedCharSetId* oder *Encoding* nicht verarbeiten kann, oder

- Wenn die integrierte Konvertierungsroutine die Daten nicht konvertieren konnte (z. B. da ein Feld oder Zeichen vorhanden ist, das nicht konvertiert werden kann).

8. Der Gültigkeitsbereich des Exits ist umgebungsabhängig. *Format*-Namen müssen so gewählt werden, dass das Risiko von Überschneidungen mit anderen Formaten minimiert ist. Es wird empfohlen, mit Zeichen zu beginnen, die die Anwendung angeben, die den Formatnamen definiert.
9. Der Datenkonvertierungsexit wird in einer Umgebung wie die des Programms ausgeführt, das den MQGET-Aufruf ausgegeben hat. Die Umgebung umfasst den Adressraum und das Benutzerprofil (sofern zutreffend). Bei dem Programm könnte es sich um einen Nachrichtenkanalagenten handeln, der Nachrichten an einen Zielwarteschlangenmanager sendet, der die Nachrichtenkonvertierung nicht unterstützt. Der Exit kann die Integrität des Warteschlangenmanagers nicht beeinträchtigen, da er nicht in der Umgebung des Warteschlangenmanagers ausgeführt wird.
10. Der einzige MQI-Aufruf, der vom Exit verwendet werden kann, ist MQXCNVC. Der Versuch, andere MQI-Aufrufe zu verwenden, schlägt mit dem Ursachencode MQRC_CALL_IN_PROGRESS oder anderen unvorhersehbaren Fehlern fehl.
11. Vom Warteschlangenmanager wird kein Einstiegspunkt mit dem Namen MQ_DATA_CONV_EXIT bereitgestellt. Jedoch wird eine `typedef` für den Namen MQ_DATA_CONV_EXIT in der Programmiersprache C bereitgestellt. Dieser Wert kann zum Deklarieren des benutzerdefinierten Exits verwendet werden, um sicherzustellen, dass die Parameter korrekt sind. Der Name des Exits muss mit dem Formatnamen (dem im Feld *Format* in MQMD enthaltenen Namen) identisch sein, auch wenn dies nicht in allen Umgebungen erforderlich ist.

    Im folgenden Beispiel ist dargestellt, wie der Exit, der das Format MYFORMAT verarbeitet, in der Programmiersprache C deklariert werden kann:

```
#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. Wenn unter z/OS ebenfalls ein API-Steuerübergabeexit in Kraft ist, wird dieser nach dem Datenkonvertierungsexit aufgerufen.

C-Aufruf

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
         InBuffer, OutBufferLength, OutBuffer);
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
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 */
```

COBOL-Deklaration (nur IBM i)

```
     CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                           INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
**   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).
```

Deklaration in System/390 Assembler

```
         CALL EXITNAME,(DATACONVEXITPARMS,MSGDESC,INBUFFERLENGTH,      X
               INBUFFER,OUTBUFFERLENGTH,OUTBUFFER)
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
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
```

Als MQRFH2-Elemente angegebene Eigenschaften

Eigenschaften, die nicht Teil des Nachrichtendeskriptors sind, können als Elemente in Ordnern des MQRFH2-Headers angegeben werden. -Übersicht über die MQRFH2-Elemente, die als Eigenschaften angegeben werden.

Dies dient zur Aufrechterhaltung der Kompatibilität mit früheren Versionen der JMS- und XMS-Clients von WebSphere MQ. In diesem Abschnitt wird beschrieben, wie Eigenschaften in MQRFH2-Headern angegeben werden.

Um MQRFH2-Elemente als Eigenschaften verwenden zu können, müssen Sie die Elemente wie im Abschnitt WebSphere MQ Classes for Java verwenden beschrieben angeben. Diese Informationen ergänzen die Informationen im Abschnitt „MQRFH2 - Header 2 für Regeln und Formatierung” auf Seite 514.

Datentypen von Zuordnungseigenschaften zu MQRFH2-Datentypen

Dieser Abschnitt enthält Informationen zu den Typen von Nachrichteneigenschaften, die ihren zugehörigen MQRFH2-Datentypen zugeordnet wurden.

Tabelle 580. Unterstützte MQRFH2-Datentypen

Nachrichteneigenschaftstyp	**MQRFH2-Datentyp**
MQBYTE[]	bin.hex
MQBOOL	boolean

Tabelle 580. Unterstützte MQRFH2-Datentypen (Forts.)	
Nachrichteneigenschaftstyp	**MQRFH2-Datentyp**
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR[]	Zeichenfolge

Bei Elementen ohne Datentyp wird der Typ "string" vorausgesetzt.

Der MQRFH2-Datentyp `int`, also eine Ganzzahl nicht angegebener Größe, wird wie `i8` behandelt.

Ein Nullwert wird durch das Elementattribut `xsi:nil='true'` angezeigt. Verwenden Sie nicht das Attribut `xsi:nil='false'` für Werte ungleich 0.

Folgende Eigenschaft hat beispielsweise einen Nullwert:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Eine Byte- oder Zeichenfolgeeigenschaft kann einen leeren Wert enthalten. Dies wird durch ein MQRFH2-Element der Länge Null dargestellt.

Folgende Eigenschaft enthält beispielsweise einen leeren Wert:

```
<EmptyProperty></EmptyProperty>
```

Unterstützte MQRFH2-Ordner

Übersicht der Verwendung von Nachrichtendeskriptorfeldern als Eigenschaften.

Die Ordner <jms>, <mcd>, <mqext>und <usr> werden im Abschnitt Header MQRFH2 und JMSbeschrieben. Der Ordner <usr> wird verwendet, um alle JMS-anwendungsdefinierten Eigenschaften zu transportieren, die einer Nachricht zugeordnet sind. Gruppen sind im Ordner <usr> nicht zulässig.

Der MQRFH2-Header und JMS unterstützt die folgenden zusätzlichen Ordner:

- <mq>

  Dieser Ordner ist für MQ-definierte Eigenschaften reserviert, die von IBM WebSphere MQ verwendet werden.

- <mq_usr>

  Dieser Ordner kann für den Transport von anwendungsdefinierten Eigenschaften verwendet werden, die nicht als benutzerdefinierte JMS-Eigenschaften verfügbar sind, weil die Eigenschaften eventuell die Anforderungen an eine JMS-Eigenschaft nicht erfüllen. Dieser Ordner kann Gruppen enthalten, die der Ordner <usr> nicht ausführen kann.

- Alle Ordner, die mit dem Attribut `content='properties'` markiert sind.

  Ein solcher Ordner entspricht dem Ordner <mq_usr> im Inhalt.

- <mqps>

  Dieser Ordner wird für Publish-/Subscribe-Eigenschaften von IBM WebSphere MQ verwendet.

IBM WebSphere MQ unterstützt außerdem die folgenden Ordner, die bereits von WAS/SIB verwendet werden:

- <sib>

  Dieser Ordner ist für die Eigenschaften von WAS-/SIB-Systemnachrichten vorgesehen und reserviert, die nicht als JMS-Eigenschaften zugänglich oder JMS_IBM_*-Eigenschaften zugeordnet, aber für WAS-/SIB-Anwendungen verfügbar sind; dazu zählen auch die Eigenschaften der Routing-Pfade vorwärts und rückwärts.

  Zumindest einige von ihnen können nicht als JMS-Eigenschaften verfügbar gemacht werden, weil es sich um Bytefeldgruppen handelt. Wenn Ihre Anwendungen Eigenschaften zu diesem Ordner hinzufügt, wird der Wert entweder ignoriert oder gelöscht.

- <sib_usr>

  Dieser Ordner ist für die Eigenschaften von WAS-/SIB-Benutzernachrichten vorgesehen und reserviert, die nicht als JMS-Benutzereigenschaften verfügbar sind, weil ihr Typ nicht unterstützt wird. Sie sind für WAS-/SIB-Anwendungen verfügbar.

  Dies sind Benutzereigenschaften, die Sie über die SIMessage-Schnittstelle abrufen oder festlegen können, jedoch wird der Inhalt der Bytefeldgruppe dem erforderlichen Eigenschaftswert zugeordnet.

  Wenn Ihre IBM WebSphere MQ-Anwendung ein beliebiges `bin.hex`-Element in den Ordner schreibt, erhält die Anwendung wahrscheinlich eine `IOException`, weil sie nicht das für die Wiederherstellung erwartete Format hat. Wenn Sie irgendetwas anderes hinzufügen als ein `bin.hex`-Element, erhalten Sie eine `ClassCastException`.

  Versuchen Sie nicht, Eigenschaften für WAS/SIB zu verwenden, indem Sie diesen Ordner verwenden; stattdessen sollten Sie den Ordner <usr> zu diesem Zweck verwenden.

- <sib_context>

  Dieser Ordner ist für die Eigenschaften von WAS-/SIB-Systemnachrichten vorgesehen, die nicht für WAS-/SIB-Benutzeranwendungen oder als JMS-Eigenschaften verfügbar sind. Dazu zählen sicherheits- und transaktionsorientierte Eigenschaften, die für Web-Service und Ähnliches verwendet werden.

  Ihre Anwendung darf keine Eigenschaften zu diesem Ordner hinzufügen.

- <mqema>

  Dieser Ordner wurde von WAS/SIB anstelle des Ordners <mqext> verwendet.

Bei MQRFH2-Ordnernamen muss die Groß- und Kleinschreibung beachtet werden.

Die folgenden Ordner, in jeder möglichen Kombination aus Groß- und Kleinbuchstaben, sind reserviert:

- Alle Ordner mit dem Präfix `mq` oder `wmq`; reserviert für IBM WebSphere MQ.
- Alle Ordner mit dem Präfix `sib`; reserviert für WAS/SIB.
- <Root> und <Body> Ordner; reserviert, aber nicht verwendet.

Die folgenden Ordner werden nicht als Ordner mit Nachrichteneigenschaften erkannt:

- <psc>

  Wird von WebSphere Message Broker verwendet, um Publish-/Subscribe-Befehlsnachrichten an den Broker zu übermitteln.

- <pscr>

  Wird von WebSphere Message Broker verwendet, um Informationen vom Broker aufzubewahren, als Reaktion auf Publish-/Subscribe-Befehlsnachrichten.

- Alle nicht von WebSphere Message Broker definierten Ordner, die nicht mit dem Attribut `content='properties'` markiert sind.

Geben Sie `content='properties'` nicht in den Ordnern <psc> oder <pscr> an. In diesem Fall würden diese Ordner wie Eigenschaften behandelt, sodass WebSphere Message Broker wahrscheinlich nicht mehr wie erwartet funktioniert.

Wenn Ihre Anwendung Nachrichten mit Eigenschaften erstellt und MQRFH2 Header als MQRFH2-Header mit Eigenschaften erkannt werden sollen, müssen die Header in der Liste der Header aufgeführt sein, die mit der Kopfzeile der Nachricht verkettet werden können.

MQRFH2 können beliebig viele MQH-Standardheader, ein MQCIH, ein MQDLH, ein MQIIH, ein MQTM, ein MQTMC2 oder ein MQXQH vorangestellt werden. Eine Zeichenfolge oder ein MQCFH beendet die Syntaxanalyse, weil sie nicht verkettet werden können.

Eine Nachricht kann mehrere MQRFH2-Header enthalten, die alle Nachrichteneigenschaften enthalten. Es können mehrere Ordner mit demselben Namen in unterschiedlichen Headern enthalten sein, falls dies nicht anderweitig z. B. von WAS/SIB eingeschränkt wird. Die Ordner werden als ein logischer Ordner erstellt, wenn sie sich alle in signifikanten Headern befinden.

Zwar können Ordner aus den signifikanten Headern nicht mit diesen Ordnern in nicht signifikanten Ordnern zusammengeführt werden, aber Ordner mit demselben Namen innerhalb der signifikanten Header können zusammengeführt werden, wobei alle widersprüchlichen Eigenschaften entfernt werden. Ihre Anwendungen dürfen nicht vom Layout von Eigenschaften innerhalb ihrer Nachricht abhängig sein.

MQRFH2-Gruppen werden für Eigenschaften in benutzerdefinierten Ordnern analysiert, d. h. nicht die Ordner: <wmq>, <jms>, <mcd>, <usr>, <mqext>, <sib>, <sib_usr>, <sib_context>und <mqema>.

Gruppen in den von IBMdefinierten Eigenschaftenordnern mit Ausnahme der Ordner <wmq> und <mq> werden für Eigenschaften geparst.

Ein MQRFH2-Ordner darf keine gemischten Inhalte enthalten. Ein Ordner oder eine Gruppe darf Gruppen, Eigenschaften oder einen Wert enthalten, aber nicht mehrere davon gleichzeitig.

Ein Segment einer Nachricht, entweder das erste oder ein folgendes Segment, darf keine anderen von IBM WebSphere MQ definierten Eigenschaften enthalten als die im Nachrichtendeskriptor beschriebenen. Deshalb schlägt die Einreihung einer Nachricht, die diese Eigenschaften enthält, wobei entweder MQMF_SEGMENT oder MQMF_SEGMENTATION_ALLOWED gesetzt ist, mit MQRC_SEGMENTATION_NOT_ALLOWED fehl.

Nachrichtengruppen hingegen können von IBM WebSphere MQ definierte Eigenschaften enthalten.

Generierung von MQRFH2-Headern

Wenn WebSphere MQ Nachrichteneigenschaften in ihre MQRFH2-Darstellung konvertiert, dann muss das Produkt MQRFH2 zur Nachricht hinzufügen. Es fügt MQRFH2 entweder als separaten Header hinzu oder führt den neuen mit einem bereits vorhandenen Header zusammen.

Die Generierung neuer MQRFH2-Header durch WebSphere MQ kann dazu führen, dass bereits vorhandene Header in einer Nachricht beschädigt werden. Bei Anwendungen, die einen Nachrichtenpuffer syntaktisch analysieren, um nach Headern zu suchen, muss berücksichtigt werden, dass die Anzahl und die Position der Header in einem Puffer unter bestimmten Umständen geändert werden. WebSphere MQ versucht, die Auswirkungen des Hinzufügens von Eigenschaften zu einer Nachricht auf ein Minimum zu begrenzen. Hierzu werden die Nachrichteneigenschaften mit einem bereits vorhandenen MQRFH2-Header zusammengeführt, wann immer dies möglich ist. Darüber hinaus versucht das Produkt, die Auswirkungen zu minimieren, indem ein generiertes MQRFH2-Element an einer festen Position relativ zu anderen Headern im Nachrichtenpuffer eingefügt wird.

Ein generierter MQRFH2-Header wird nach dem Element MQMD und nach einer beliebigen Anzahl von MQXQH-, MQRFH- und MQDLH-Headern platziert, deren Reihenfolge keine Rolle spielt. Der generierte MQRFH2-Header wird direkt vor dem ersten Header platziert, bei dem es sich nicht um einen MQMD-, MQXQH-, MQDLH- oder MQRFH-Header handelt.

Regeln zum Zusammenführen generierter MQRFH2-Header

Die folgenden Regeln gelten für das Zusammenführen eines generierten MQRFH2-Headers mit einem bereits vorhandenen MQRFH2-Header. Der generierte MQRFH2-Header wird mit einem bereits vorhandenen MQRFH2-Header zusammengeführt, wenn die folgenden Bedingungen gelten:

1. Der vorhandene MQRFH2-Header befindet sich an derselben Position, unter der WebSphere MQ einen generierten MQRFH2-Header platzieren würde, oder aber er befindet sich an einer früheren Position innerhalb der Headerkette.
2. Die CCSID (ID des codierten Zeichensatzes) der generierten Eigenschaften ist identisch mit dem Wert von NameValueCCSID des vorhandenen MQRFH2-Headers.

Andernfalls wird der generierte Header unter der zuvor beschriebenen Position separat in den Puffer gestellt.

Regeln zum Zusammenführen von Ordnern in einem vorhandenen MQRFH2-Header

Wenn Nachrichteneigenschaften in einem bereits vorhandenen MQRFH2-Header zusammengeführt werden, dann wird der vorhandene MQRFH2-Header auf Ordner durchsucht, die mit den Nachrichteneigenschaften übereinstimmen. Anschließend wird die Zusammenführung durchgeführt. Wenn kein übereinstimmender Ordner vorhanden ist, dann wird am Ende des vorhandenen Ordners ein neuer Ordner hinzugefügt. Wenn ein übereinstimmender Ordner vorhanden ist, dann wird der Ordner durchsucht. Eventuell vorhandene, übereinstimmende Eigenschaften werden überschrieben. Neue Eigenschaften werden am Ende des Ordners hinzugefügt.

Einschränkungen für MQRFH2-Ordner

Überschrift der Ordnereinschränkungen in MQRFH2-Headern

Die MQRFH2-Einschränkungen gelten für folgende Ordner:

- Elementnamen im Ordner <usr> dürfen nicht mit dem Präfix JMSbeginnen. Diese Eigenschaftsnamen sind für JMS reserviert und für benutzerdefinierte Eigenschaften nicht gültig.

  Ein solcher Elementname führt nicht dazu, dass die Syntaxanalyse von MQRFH2 fehlschlägt, aber er ist für die APIs der WebSphere MQ-Nachrichteneigenschaften nicht zugänglich.
- Elementnamen im Ordner <usr> dürfen in keiner Mischung aus Groß- oder Kleinschreibung, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS und ESCAPE sein. Diese Namen stimmen mit SQL-Schlüsselwörtern überein und machen die Syntaxanalyse von Selektoren schwieriger, da <usr> der Standardordner ist, der verwendet wird, wenn für eine bestimmte Eigenschaft in einem Selektor kein Ordner angegeben ist.

  Ein solcher Elementname führt nicht dazu, dass die Syntaxanalyse von MQRFH2 fehlschlägt, aber er ist für die APIs der WebSphere MQ-Nachrichteneigenschaften nicht zugänglich.
- Elementnamen in einem beliebigen Ordner, der Nachrichteneigenschaften enthalten soll, dürfen keinen Punkt (.) enthalten. (Unicode-Zeichen U+002E), da dies in Eigenschaftsnamen zur Angabe der Hierarchie verwendet wird.

  Ein solcher Elementname führt nicht dazu, dass die Syntaxanalyse von MQRFH2 fehlschlägt, aber er ist für die APIs der WebSphere MQ-Nachrichteneigenschaften nicht zugänglich.

Grundsätzlich können MQRFH2-Header, die gültige Daten im XML-Stil enthalten, von WebSphere MQ ohne Fehler abgefragt werden, obwohl bestimmte Elemente von MQRFH2 nicht über die APIs der WebSphere MQ-Nachrichteneigenschaften zugänglich sind.

Namenskonflikte bei MQRFH2

Übersicht der Konflikte innerhalb von MQRFH2-Elementnamen.

Einer Nachrichteneigenschaft kann nur ein Wert zugeordnet werden. Hat ein Zugriffsversuch auf eine Eigenschaft einen Konflikt der Werte zur Folge, wird einem von ihnen der Vorzug gegenüber dem anderen gegeben.

Die WebSphere MQ-Syntax zum Zugriff auf MQRFH2-Elemente ermöglicht eine eindeutige Identifizierung eines Elements, wenn ein Ordner keine Elemente mit demselben Namen enthält. Enthält ein Ordner mehrere Elemente mit demselben Namen, wird für die Eigenschaft der Wert verwendet, der am nächsten bei der Kopfzeile der Nachricht steht.

Dies gilt, wenn zwei oder mehr Ordner desselben Namens in unterschiedlichen signifikanten MQRFH2-Headern in derselben Nachricht enthalten sind.

Ein Konflikt kann auftreten, wenn ein MQGET-Aufruf verarbeitet wird, nachdem eine andere Eigenschaft als ein Nachrichtendeskriptor zweimal gesetzt wurde: sowohl über einen MQSETMP-Aufruf als auch direkt im unformatierten MQRFH2-Header.

In diesem Fall hat die der Nachricht von einem API-Aufruf zugeordnete Eigenschaft vor einer Eigenschaft in den Nachrichtendaten Vorrang, also der Eigenschaft im unformatierten MQRFH2-Header. Bei einem Konflikt wird davon ausgegangen, dass sie logisch vor den Nachrichtendaten steht.

Eigenschaftsnamen zu MQRFH2-Ordner- und -Elementnamen zuordnen

Übersicht über die Unterschiede zwischen Eigenschaftsnamen und Elementnamen im MQRFH2-Header.

Werden definierte APIs, die letztendlich MQRFH2-Header generieren, zur Angabe von Eigenschaftsnamen verwendet (z. B. MQ JMS), ist der Eigenschaftsname nicht notwendigerweise der Elementname im MQRFH2-Ordner.

Deshalb wird der Eigenschaftsname dem MQRFH2-Element zugeordnet und umgekehrt, wobei sowohl der Ordnername, der das Element enthält, als auch der Elementname berücksichtigt wird. Einige Beispiele aus IBM WebSphere MQ classes for JMS sind bereits in Java verwendendokumentiert.

Eigenschaftsname	MQRFH2-Ordnername	MQRFH2-Elementname
`JMSDestination`	`jms`	`Dst`
`JMSType`	`mcd`	`Type`, `Set`, `Fmt`
`xxx` (benutzerdefiniert, wobei `xxx` nicht mit JMS beginnt)	`usr`	`xxx`

Wenn also eine JMS-Anwendung auf die Eigenschaft `JMSDestination` zugreift, wird dies dem Element `Dst` im Ordner <jms> zugeordnet.

Bei der Angabe von Eigenschaften als MQRFH2 -Elemente definiert IBM WebSphere MQ seine Elemente wie folgt:

Eigenschaftsname	MQRFH2-Ordnername	MQRFH2-Gruppenname	MQRFH2-Elementname
<Property>	<usr>	Nicht zutreffend	<Property>
<folder>.<Property>	<folder>	Nicht zutreffend	<Property>
<folder>.<group>.<Property>	<folder>	<group>	<Property>

Wenn eine IBM WebSphere MQ -JMS-Anwendung beispielsweise versucht, auf die Eigenschaft `Property1` zuzugreifen, wird dies dem Element `Property1` im Ordner <usr> zugeordnet. Die Eigenschaft `wmq.Property2` wird der Eigenschaft `Property2` im Ordner <wmq> zugeordnet.

Enthält der Eigenschaftsname mehr als ein Zeichen ".", wird der MQRFH2-Elementname nach dem abschließenden "." -Zeichen verwendet und mithilfe von MQRFH2-Gruppen wird eine Hierarchie gebildet. Verschachtelte MQRFH2-Gruppen sind zulässig.

Der Zugriff auf die JMS-Header-und providerspezifischen Eigenschaften, die in einem MQRFH2 in den Ordnern <mcd>, <jms>und <mqext> enthalten sind, erfolgt über eine IBM WebSphere MQ -Anwendung unter Verwendung der Kurznamen, die in Using WebSphere MQ Classes for Javadefiniert sind.

Auf benutzerdefinierte JMS-Eigenschaften wird über den Ordner <usr> zugegriffen. Eine IBM WebSphere MQ -Anwendung kann den Ordner <usr> für ihre Anwendungseigenschaften verwenden, wenn es akzeptabel ist, dass die Eigenschaft JMS-Anwendungen als eine ihrer benutzerdefinierten Eigenschaften angezeigt wird.

Wenn dies nicht akzeptabel ist, wählen Sie einen anderen Ordner aus. Der Ordner <wmq_usr> wird als Standardposition für solche Nicht-JMS-Eigenschaften bereitgestellt.

Ihre Anwendungen können jeden beliebigen MQRFH2-Ordner mit klar strukturierter Verwendung angeben und verwenden, der nicht in „Als MQRFH2-Elemente angegebene Eigenschaften” auf Seite 932 dokumentiert ist, wenn Sie Folgendes beachten:

1. Der Ordner könnte bereits in Gebrauch sein oder in Zukunft von einer anderen Anwendung verwendet werden, die einen undefinierten Zugriff auf die darin enthaltenen Eigenschaften gewährt. Die vorgeschlagenen Namenskonventionen für Eigenschaftsnamen finden Sie unter Eigenschaftsnamen.
2. Die Eigenschaften sind nicht für frühere Versionen des IBM WebSphere MQ classes for JMS -oder XMS -Clients zugänglich, die nur auf den Ordner <usr> für benutzerdefinierte Eigenschaften zugreifen können.
3. Der Ordner muss mit dem Attribut `content` markiert werden, dessen Wert auf `properties` gesetzt ist, zum Beispiel `content='properties'`.

   „MQSETMP - Nachrichteneigenschaft festlegen” auf Seite 778 fügt dieses Attribut automatisch nach Bedarf hinzu. Dieses Attribut darf keinem der von IBMdefinierten Ordner hinzugefügt werden, beispielsweise <jms> und <usr>. Dadurch wird die Nachricht vom IBM WebSphere MQ classes for JMS -Client vor Version 7.0mit einem `MessageFormatException`zurückgewiesen.

Da der Ordner <usr> die Standardposition für Eigenschaften der Syntax <Property> ist, verwenden eine IBM WebSphere MQ -Anwendung und eine JMS-Anwendung denselben Namen, um auf denselben benutzerdefinierten Eigenschaftswert zuzugreifen.

Reservierte Ordnernamen

Es gibt mehrere reservierte Ordnernamen. Sie können keine Namen wie zum Beispiel Ihre Ordnerpräfixe verwenden. So kann beispielsweise `Root.Property1` nicht auf eine gültige Eigenschaft zugreifen, weil `Root` reserviert ist. Die folgende Liste enthält reservierte Ordnernamen:

- Root
- Hauptteil
- Eigenschaften
- Umgebung
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

Eigenschaftsdeskriptorfelder auf MQRFH2-Header abbilden

Wenn eine Eigenschaft in ein MQRFH2-Element umgesetzt wird, werden die folgenden Elementattribute verwendet, um die signifikanten Felder des Eigenschaftendeskriptors anzugeben: In diesem Abschnitt wird beschrieben, wie MQPD-Felder in Elementattribute von MQRFH2 umgesetzt werden.

Support

Das Eigenschaftsdeskriptorfeld "Support" ist in drei Elementattribute aufgeteilt

- Das Elementattribut **sr** gibt Werte in der Bitmaske MQPD_REJECT_UNSUP_MASK an.
- Das Elementattribut **sa** gibt Werte in der Bitmaske MQPD_ACCEPT_UNSUP_MASK an.
- Das Elementattribut **sx** gibt Werte in der Bitmaske MQPD_ACCEPT_UNSUP_IF_XMIT_MASK an.

Diese Elementattribute gelten nur im Ordner <mq>. Sie werden ignoriert, wenn sie für Elemente in anderen Ordnern mit Eigenschaften gesetzt werden.

Support-Wert	MQRFH2-Elementattribut	MQRFH2-Attributwert
MQPD_SUPPORT_OPTIONAL	sa	optional Dies ist der Standardwert.
MQPD_SUPPORT_REQUIRED	sr	Erforderlich
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	Lokal

Context

Das Elementattribut **context** zeigt an, zu welchem Nachrichtenkontext eine Eigenschaft gehört. Verwenden Sie nur einen Wert. Dieses Elementattribut gilt für Eigenschaften in jedem beliebigen Ordner, der Eigenschaften enthält.

Context-Wert	MQRFH2-Attributwert
MQPD_NO_CONTEXT	none Dies ist der Standardwert.
MQPD_USER_CONTEXT	Benutzer

CopyOptions

Verwenden Sie das Elementattribut **copy**, um Nachrichten anzugeben, in die eine Eigenschaft kopiert werden soll. Es ist mehr als ein Wert zulässig; trennen Sie mehrere Werte durch ein Komma. Zum Beispiel sind **copy='reply'** und **copy='publish,report'** beide gültig. Dieses Elementattribut gilt für Eigenschaften in jedem beliebigen Ordner, der Eigenschaften enthält.

Anmerkung: In der Attributdefinition sind einzelne oder doppelte Anführungszeichen gültig, zum Beispiel **copy='reply'** oder **copy="report"**

CopyOption-Wert	MQRFH2-Attributwert
MQPD_COPY_FORWARD	forward
MQPD_COPY_REPLY	reply
MQPD_COPY_REPORT	Bericht
MQPD_COPY_PUBLISH	veröffentlichen
MQPD_COPY_ALL	alle Geben Sie dies nicht zusammen mit anderen Werten an. Wenn es mit einem anderen Wert verwendet wird, hat es vor allen Werten Vorrang außer **none**.

CopyOption-Wert	MQRFH2-Attributwert
MQPD_COPY_DEFAULT	Standard Dies ist der Standardwert. Es entspricht der Angabe der drei Werte MQCOPY_FORWARD, MQCOPY_REPORT und MQCOPY_PUBLISH. Geben Sie dies nicht zusammen mit anderen Werten an.
MQPD_COPY_NONE	none Geben Sie dies nicht zusammen mit anderen Werten an. Wenn es mit einem anderen Wert verwendet wird, hat es Vorrang.

Einschränkungen für den Ordner <mq> MQRFH2

Wenn eine Nachricht in eine Warteschlange gestellt wird, wird sie nach dem Ordner <mq> durchsucht, damit die Nachricht entsprechend ihren MQ-definierten Eigenschaften verarbeitet werden kann. Um eine effiziente Syntaxanalyse MQ-definierter Eigenschaften zu ermöglichen, gelten für den Ordner folgende Einschränkungen:

- MQ berücksichtigt nur Eigenschaften im ersten bedeutsamen <mq>-Ordner in der Nachricht; Eigenschaften in allen anderen <mq>-Ordnern in der Nachricht werden ignoriert.
- Wenn der Ordner für UTF-8 festgelegt ist, darf der Ordner nur UTF-8-Einzelbytezeichen enthalten. Ein Mehrfachbytezeichen im Ordner kann dazu führen, dass die Syntaxanalyse fehlschlägt und die Nachricht abgelehnt wird.
- Schließen Sie keine MQRFH2-Gruppen in einen <mq>-Ordner ein. Das Unicode-Zeichen U+003C in einem Eigenschaftswert führt dazu, dass die Nachricht abgelehnt wird.
- Im Ordner sollten keine Escapezeichenfolgen verwendet werden. Eine Escapezeichenfolge wird wie der eigentliche Wert des Elements behandelt.
- Nur das Unicode-Zeichen U+0020 wird als Leerzeichen innerhalb des Ordners behandelt. Alle anderen Zeichen werden als bedeutsam betrachtet und können dazu führen, dass die Syntaxanalyse fehlschlägt und die Nachricht abgelehnt wird.

Wenn die Syntaxanalyse des < mq> -Ordners fehlschlägt oder wenn der Ordner diese Einschränkungen nicht beachtet, wird die Nachricht mit CompCode **MQCC_FAILED** und Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**zurückgewiesen.

Ungültige MQRFH2-Header

Während der Verarbeitung eines MQPUT-, MQPUT1- oder MQGET-Aufrufs kann eine teilweise Analyse der MQRFH2-Header in der Nachricht stattfinden, um zu überprüfen, welche Ordner eingeschlossen sind und um festzustellen, ob die Ordner Eigenschaften enthalten. -Übersicht der MQRFH2-Header, die nicht gültig sind.

Kann die partielle Analyse der Nachricht nicht erfolgreich beendet werden, weil die Struktur ungültig ist, zum Beispiel wenn das Feld `StrucLength` zu klein ist, tritt Folgendes auf:

- Der Aufruf MQPUT oder MQPUT1 schlägt mit dem Ursachencode MQRC_RFH_ERROR fehl, wenn festgestellt werden kann, dass die Anwendung eine Option von WebSphere MQ Version 7 enthält, sodass vorhandene Anwendungen nicht fehlschlagen.
- Der Aufruf MQGET wird erfolgreich zurückgemeldet und MQRFH2 mit dem Fehler wird in den angegebenen Puffer zurückgemeldet.

Wenn die partielle Syntaxanalyse fehlschlägt, weil nicht festgestellt werden kann, ob ein bestimmter Ordner Eigenschaften enthält oder nicht, z. B. wenn der Ordner `<<jms` beginnt, schlägt die Syntaxanalyse fehl, bevor der Ordnername bestimmt wird.

- Der Aufruf MQPUT oder MQPUT1 schlägt mit dem Ursachencode MQRC_RFH_FORMAT_ERROR fehl, wenn festgestellt werden kann, dass die Anwendung eine Option von WebSphere MQ Version 7 enthält, sodass vorhandene Anwendungen nicht fehlschlagen.
- Der Aufruf MQGET wird erfolgreich zurückgemeldet und MQRFH2 mit dem Fehler wird in den angegebenen Puffer zurückgemeldet.
- Intern im Warteschlangenmanager wird die Nachricht zwar nicht wegen des schlecht formatierten Ordners zurückgewiesen, aber der Ordner wird grundsätzlich so behandelt, als ob er keine Eigenschaften enthalten würde.

Eine Nachricht kann das Warteschlangenmanagernetz mit einem Ordner mit einem solchen Syntaxfehler durchlaufen, aber diese werden niemals analysiert und erkannt, wenn ein oder mehrere Ordner in der Nachricht:

- gültig sind
- erfolgreich analysiert wurden
- bei der Verarbeitung der Nachricht verwendet werden.

Daher ist nicht garantiert, dass sie erkannt werden.

Wenn eine Ihrer Anwendungen „MQSETMP - Nachrichteneigenschaft festlegen” auf Seite 778 oder MQINQMP zum Zugriff auf eine Eigenschaft verwendet und dadurch eine vollständige Analyse eines MQRFH2-Ordners ausgelöst wird, wobei ein Fehler erkannt wird, durch den die Analyse nicht abgeschlossen werden kann, wird dies durch einen entsprechenden Rückkehrcode für den API-Aufruf angezeigt. Der Anwendung werden keine Eigenschaften im Ordner verfügbar gemacht.

Wenn der Parser beim Versuch einer vollständigen Analyse eines MQRFH2-Ordners unerkannte Elementattribute oder einen unbekannten Datentyp findet, wird die Analyse fortgesetzt und erfolgreich abgeschlossen, ohne dass eine Warnung ausgegeben wird, weil dies keinen Analysefehler darstellt.

Codepagekonvertierung

In diesem Abschnitt wird die Unterstützung von Namen und IDs von codierten Zeichensätzen, Landessprache, z/OS-Konvertierung, IBM i-Konvertierung und Unicode-Konvertierung beschrieben.

In jedem Landessprachenabschnitt werden folgende Informationen aufgelistet:

- Native IDs des codierten Zeichensatzes, die unterstützt werden
- Codepagekonvertierungen, die **nicht** unterstützt werden

In den Informationen werden folgende Begriffe verwendet:

-8
Bedeutet für HP-UX, dass es die ID des codierten Zeichensatzes für den in HP-UX definierten, codierten Zeichensatz *roman8* ist

AIX
Bedeutet WebSphere MQ for AIX

HP-UX
Bedeutet WebSphere MQ for HP-UX

Linux
Gibt WebSphere MQ für Linux für Intel und WebSphere MQ für Linux für zSeries an

HP Integrity NonStop Server
Bedeutet WebSphere MQ for HP Integrity NonStop Server

OS/400
Bedeutet WebSphere MQ for IBM i

Solaris
Bedeutet WebSphere MQ for Solaris

Windows
Bedeutet WebSphere MQ for Windows

z/OS
Bedeutet WebSphere MQ for z/OS

Gemäß Standardeinstellung für die Datenkonvertierung wird die Konvertierung für das Zielsystem (empfangende System) durchgeführt.

Wenn das Quellenprodukt die Konvertierung unterstützt, können Sie einen Kanal einrichten und Daten austauschen, indem Sie das Kanalattribut CONVERT an der Quelle auf YES setzen.

Anmerkung:

1. Die Konvertierung von Informationen des MQI-Clients von WebSphere MQ findet im Server statt, d. h., der Server muss die Konvertierung von der ID des codierten Zeichensatzes des Clients in die ID des codierten Zeichensatzes des Servers unterstützen.
2. Die Konvertierung kann die Unterstützung einschließen, die durch die CSD/PTF zur neuesten Version von WebSphere MQ hinzugefügt wurde. Überprüfen Sie den Inhalt des neuesten Service-Levels daraufhin, ob Sie eine CSD/PTF installieren müssen, um diese Konvertierung zu ermöglichen.

Der Abschnitt Tabelle 581 auf Seite 942 enthält einen Querverweis zwischen einigen der Nummern von IDs des codierten Zeichensatzes und einigen branchenspezifischen Namen von codierten Zeichensätzen.

Namen von codierten Zeichensätzen und CCSIDs

WebSphere MQ for z/OS stellt eine umfassendere Konvertierung bereit als in den sprachenspezifischen Tabellen aufgeführt ist.

Tabelle 581. Namen von codierten Zeichensätzen und CCSIDs

Namen von codierten Zeichensätzen	CCSIDs
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (Euro)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
PCK	943
GBK	1386
koi8-r	878

Landessprachen

Diese Informationen enthalten Sprachen, die von WebSphere MQ unterstützt werden.

Folgende Sprachen werden von WebSphere MQ unterstützt:

- Amerikanisches Englisch - siehe „Amerikanisches Englisch” auf Seite 943
- Deutsch - siehe „German” auf Seite 944
- Dänisch und Norwegisch - siehe „Dänisch und Norwegisch” auf Seite 944
- Finnisch und Schwedisch - siehe „Finnisch und Schwedisch” auf Seite 945
- Italienisch - siehe „Italian” auf Seite 946
- Spanisch - siehe „Spanish” auf Seite 947
- Britisches Englisch/Gälisch - siehe „Britisches Englisch/Gälisch” auf Seite 947
- Französisch - siehe „French” auf Seite 948
- Mehrsprachig - siehe „Mehrsprachig” auf Seite 948
- Portugiesisch - siehe „Portugiesisch” auf Seite 949
- Isländisch - siehe „Isländisch” auf Seite 950
- Osteuropäische Sprachen - siehe „Osteuropäische Sprachen” auf Seite 951
- Kyrillisch - siehe „Kyrillisch” auf Seite 952
- Estnisch - siehe „Estnisch” auf Seite 952
- Lettisch und Litauisch - siehe „Lettisch und Litauisch” auf Seite 953
- Ukrainisch - siehe „Ukrainisch” auf Seite 955
- Griechisch - siehe „Griechisch” auf Seite 955
- Türkisch - siehe „Türkisch” auf Seite 956
- Hebräisch - siehe „Hebräisch” auf Seite 956
- Farsi - siehe „Farsi” auf Seite 958
- Urdu - siehe „Urdu” auf Seite 958
- Thailändisch - siehe „Thailändisch” auf Seite 959
- Laotisch - siehe „Laotisch” auf Seite 959
- Vietnamesisch - siehe „Vietnamesisch” auf Seite 959
- Japanisch mit lateinischem Einzelbytezeichensatz - siehe „Japanisch mit lateinischem Einzelbytezeichensatz” auf Seite 960
- Japanisch mit Katakana-Einzelbytezeichensatz - siehe „Japanisch, Katakana-Einzelbytezeichensatz” auf Seite 961
- Japanisch mit Mischung aus Kanji/Lateinisch - siehe „Japanisch mit Mischung aus Kanji/Latein” auf Seite 963
- Japanisch mit aus Kanji/Katakana - siehe „Japanisch, Kanji/Katakana gemischt” auf Seite 964
- Koreanisch - siehe „Korean” auf Seite 966
- Vereinfachtes Chinesisch - siehe „Vereinfachtes Chinesisch” auf Seite 966
- Traditionelles Chinesisch - siehe „Traditionelles Chinesisch” auf Seite 967

Amerikanisches Englisch

Details zu CCSIDs und CCSID-Konvertierung für amerikanisches Englisch.

In der folgenden Tabelle sind die nativen CCSIDs für amerikanisches Englisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	37, 924, 1140

Plattform	Native CCSIDs
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 1252, 5348, 858
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

273
Keine Konvertierung in Codepages 923, 858

924
Keine Konvertierung in Codepages 437, 858, 1051, 1140, 1252, 1275, 5348

1140
Keine Konvertierung in Codepages 924, 1051, 1275

German

Details zu CCSIDs und CCSID-Konvertierung für Deutsch.

In der folgenden Tabelle sind die nativen CCSIDs für Deutsch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	273, 924, 1141
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

273
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141
Keine Konvertierung in Codepages 924, 1051, 1275

Dänisch und Norwegisch

Details zu CCSIDs und CCSID-Konvertierung für Dänisch und Norwegisch.

In der folgenden Tabelle sind die nativen CCSIDs für Dänisch und Norwegisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	277, 924, 1142
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 865, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

277
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142
Keine Konvertierung in Codepages 924, 865, 1051, 1275

AIX

Codepage:

819
Keine Konvertierung in Codepage 865

HP-UX

Codepage:

1051
Keine Konvertierung in Codepage 865

Windows

Codepage:

865
Keine Konvertierung in Codepages 1051, 1275

Finnisch und Schwedisch

Details zu CCSIDs und CCSID-Konvertierung für Finnisch und Schwedisch.

In der folgenden Tabelle sind die nativen CCSIDs für Finnisch und Schwedisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	278, 924, 1143
AIX	819, 923, 5348

Plattform	Native CCSIDs
HP-UX	819, 923, 1051
Windows	437, 850, 858, 865, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

278
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143
Keine Konvertierung in Codepages 865, 924, 1051, 1275

AIX

Codepage:

819
Keine Konvertierung in Codepage 865

850
Keine Konvertierung in Codepage 865

HP-UX

Codepage:

1051
Keine Konvertierung in Codepage 865

Windows

Codepage:

865
Keine Konvertierung in Codepages 1051, 1275

Italian

Details zu CCSIDs und CCSID-Konvertierung für Italienisch.

In der folgenden Tabelle sind die nativen CCSIDs für Italienisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	280, 924, 1144
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923

Plattform	Native CCSIDs
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

280
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144
Keine Konvertierung in Codepages 924, 1051, 1275

Spanish

Details zu CCSIDs und CCSID-Konvertierung für Spanisch.

In der folgenden Tabelle sind die nativen CCSIDs für Spanisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	284, 924, 1145
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

284
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145
Keine Konvertierung in Codepages 924, 1051, 1275

Britisches Englisch/Gälisch

Details zu CCSIDs und CCSID-Konvertierung für britisches Englisch/Gälisch.

In der folgenden Tabelle sind die nativen CCSIDs für britisches Englisch/Gälisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	285, 924, 1146
AIX	819, 923, 5348

Plattform	Native CCSIDs
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

285
Keine Konvertierung in Codepages 858, 923, 924, 1275

924
Keine Konvertierung in Codepages 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146
Keine Konvertierung in Codepages 924, 1051, 1275

French

Details zu CCSIDs und CCSID-Konvertierung für Französisch.

In der folgenden Tabelle sind die nativen CCSIDs für Französisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	297, 924, 1147
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

297
Keine Konvertierung in Codepages 858, 923, 924, 1275, 5348

924
Keine Konvertierung in Codepages 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147
Keine Konvertierung in Codepages 924, 1051, 1275

Mehrsprachig

Details zu CCSIDs und CCSID-Konvertierung für Mehrsprachig.

In der folgenden Tabelle sind die nativen CCSIDs für die mehrsprachige Konvertierung auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	500, 924, 1148
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, SINIX, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

500
Keine Konvertierung in Codepages 858, 923

924
Keine Konvertierung in Codepages 437, 858, 1051, 1148, 1252, 1275, 5348

1148
Keine Konvertierung in Codepages 924, 1051, 1275

Portugiesisch

Details zu CCSIDs und CCSID-Konvertierung für Portugiesisch.

In der folgenden Tabelle sind die nativen CCSIDs für Portugiesisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i	37, 500, 924, 1140
z/OS	500, 924, 1140
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 860, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

273
Keine Konvertierung in Codepages 858, 923, 1275

500
Keine Konvertierung in Codepages 858, 923, 1275

924
Keine Konvertierung in Codepages 858, 860, 1051, 1140, 1252, 1275, 5348

1140
Keine Konvertierung in Codepages 860, 924, 1051, 1275

HP-UX

Codepage:

1051
Keine Konvertierung in Codepage 860

Windows

Codepage:

860
Keine Konvertierung in Codepages 1051, 1275

Isländisch

Details zu CCSIDs und CCSID-Konvertierung für Isländisch.

In der folgenden Tabelle sind die nativen CCSIDs für Isländisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	871, 924, 1149
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 861, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Apple-Client	1275

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

871
Keine Konvertierung in Codepages 858, 923, 924, 1275, 5348

924
Keine Konvertierung in Codepages 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149
Keine Konvertierung in Codepages 924, 1051, 1275

HP-UX

Codepage:

1051
Keine Konvertierung in Codepage 861

Windows

Codepage:

861
Keine Konvertierung in Codepages 1051, 1275

Osteuropäische Sprachen

Details zu CCSIDs und CCSID-Konvertierung für osteuropäische Sprachen. Zu den typischen Sprachen, die diese CCSIDs verwenden, gehören Albanisch, Kroatisch, Tschechisch, Ungarisch, Polnisch, Rumänisch, Serbisch, Slowakisch und Slowenisch.

In der folgenden Tabelle sind die nativen CCSIDs für osteuropäische Sprachen auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	870, 1153
Windows	852, 1250, 5346, 9044
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	912
Osteuropäischer Apple-Client	1282
Rumänischer Apple-Client	1285
Kroatischer Apple-Client	1284

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

870
Keine Konvertierung in Codepages 1284, 1285

1153
Keine Konvertierung in Codepages 1250, 1284, 1285

IBM i

Codepage:

870
Keine Konvertierung in Codepages 1284, 1285, 5346, 9044

1153
Keine Konvertierung in Codepages 1282, 1284, 1285, 5346, 9044

HP-UX, Solaris, Linux

Codepage:

912
Keine Konvertierung in Codepages 1284, 1285

HP Integrity NonStop Server

Codepage:

912
Keine Konvertierung in Codepages 1153, 1284, 1285, 9044

Windows

Codepage:

852
Keine Konvertierung in Codepages 1284, 1285

1250
Keine Konvertierung in Codepages 1284, 1285

9044
Keine Konvertierung in Codepages 912, 1282, 1284, 1285

Kyrillisch

Details zu CCSIDs und CCSID-Konvertierung für Kyrillisch. Zu den typischen Sprachen, die diese CCSIDs verwenden, gehören Bulgarisch, Mazedonisch, Russisch, Serbisch und Weißrussisch.

In der folgenden Tabelle sind die nativen CCSIDs für Kyrillisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
z/OS	1025
IBM i	880, 1025
Windows	855, 866, 1131, 1251, 5347
Solaris	878, 915
AIX, HP-UX, Linux, HP Integrity NonStop Server	915
Apple-Client	1283

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

880
Keine Konvertierung in Codepages 855, 866, 878, 1131, 5347

1025
Keine Konvertierung in Codepages 878, 5347

Windows

Codepage:

855
Keine Konvertierung in Codepage 1131

866
Keine Konvertierung in Codepage 1131

1131
Keine Konvertierung in Codepages 855, 866, 880, 1283

Estnisch

Details zu CCSIDs und CCSID-Konvertierung für Estnisch.

In der folgenden Tabelle sind die nativen CCSIDs für Estnisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1122, 1157

Plattform	Native CCSIDs
Windows	902, 922, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	902, 922
HP Integrity NonStop Server	922

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

1122
Keine Konvertierung in Codepages 902, 1157, 9449

1157
Keine Konvertierung in Codepages 922, 1122, 1257, 9449

IBM i

Codepage:

1122
Keine Konvertierung in Codepages 902, 5353, 9449

1157
Keine Konvertierung in Codepages 922, 5353, 9449

HP-UX, Solaris, Linux

Codepage:

902
Keine Konvertierung in Codepages 922, 1122, 9449

922
Keine Konvertierung in Codepages 902, 1157, 9449

Windows

Codepage:

5353
Keine Konvertierung in Codepage 9449

9449
Keine Konvertierung in Codepages 902, 922, 1122, 1157, 1257, 5353

902
Keine Konvertierung in Codepages 922, 1122, 9449

HP Integrity NonStop Server

Codepage:

922
Keine Konvertierung in Codepages 902, 1157, 9449

Lettisch und Litauisch

Details zu CCSIDs und CCSID-Konvertierung für Lettisch und Litauisch.

In der folgenden Tabelle sind die nativen CCSIDs für Lettisch und Litauisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1112, 1156
Windows	901, 921, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	901, 921
HP Integrity NonStop Server	921

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

1112
Keine Konvertierung in Codepages 901, 1156, 9449

1156
Keine Konvertierung in Codepages 901, 1156, 9449

IBM i

Codepage:

1112
Keine Konvertierung in Codepage 5353

1153
Keine Konvertierung in Codepages 921, 5353, 9449

HP-UX, Solaris, Linux

Codepage:

902
Keine Konvertierung in Codepages 921, 1112, 1257, 9449

921
Keine Konvertierung in Codepages 901, 1156, 9449

Windows

Codepage:

901
Keine Konvertierung in Codepages 921, 1112, 1257, 9449

5355
Keine Konvertierung in Codepage 9449

9449
Keine Konvertierung in Codepages 901, 921, 1112, 1156, 1257

HP Integrity NonStop Server

Codepage:

921
Keine Konvertierung in Codepages 901, 1156, 9449

Ukrainisch

Details zu CCSIDs und CCSID-Konvertierung für Ukrainisch.

In der folgenden Tabelle sind die nativen CCSIDs für Ukrainisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1123
Windows	1124, 1125, 1251, 5347
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1124

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

1123
Keine Konvertierung in Codepage 5347

HP-UX

Codepage:

1124
Keine Konvertierung in Codepage 5347

Windows

Codepage:

1125
Keine Konvertierung in Codepage 1123

Griechisch

Details zu CCSIDs und CCSID-Konvertierung für Griechisch.

In der folgenden Tabelle sind die nativen CCSIDs für Griechisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	875
HP-UX	813 (siehe Hinweis)
Windows	869, 1253, 5349
AIX, NCR, HP Integrity NonStop Server, Solaris, Linux	813
Apple-Client	1280
DOS-Client	737
Anmerkung: Unter HP-UX wird nur der codierte ISO-Zeichensatz unterstützt. Der HP-UX-proprietäre codierte Zeichensatz greek8 hat keine registrierte CCSID und wird nicht unterstützt.	

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

875
Keine Konvertierung in Codepage 5349

Windows

Codepage:

1253
Keine Konvertierung in Codepage 737

5349
Keine Konvertierung in Codepage 737

Türkisch

Details zu CCSIDs und CCSID-Konvertierung für Türkisch.

In der folgenden Tabelle sind die nativen CCSIDs für Türkisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1026
HP-UX	920 (siehe Hinweis)
Windows	857, 1254, 5350
AIX, HP Integrity NonStop Server, Solaris, Linux	920
Apple-Client	1281
Anmerkung: Unter HP-UX wird nur der codierte ISO-Zeichensatz unterstützt. Der HP-UX-proprietäre codierte Zeichensatz turkish8 hat keine registrierte CCSID und wird nicht unterstützt.	

Alle Nicht-Clientplattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs der anderen Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

1026
Keine Konvertierung in Codepage 5350

Hebräisch

Details zu CCSIDs und CCSID-Konvertierung für Hebräisch.

In der folgenden Tabelle sind die nativen CCSIDs für Hebräisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
z/OS	424, 803, 4899, 12712
IBM i	424
AIX	916, 9048
HP-UX	916 (siehe Hinweis)
Windows	1255, 5351
HP Integrity NonStop Server, Solaris, Linux	916

Plattform	Native CCSIDs
Anmerkung: Unter HP-UX wird nur der codierte ISO-Zeichensatz unterstützt. Der HP-UX-proprietäre codierte Zeichensatz greek8 hat keine registrierte CCSID und wird nicht unterstützt.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

424
Keine Konvertierung in Codepages 867, 4899, 9048, 12712

803
Keine Konvertierung in Codepages 867, 4899, 5351, 9048, 12712

4899
Keine Konvertierung in Codepages 424, 803, 856, 862, 916, 1255

12712
Keine Konvertierung in Codepages 424, 803, 856, 916, 1255

IBM i

Codepage:

424
Keine Konvertierung in Codepages 803, 867, 4899, 5351, 9048, 12712

Codepage 424 wird ebenfalls in und aus CCSID 4952 konvertiert. Dabei handelt es sich um eine Variante von 856.

AIX

Codepage:

916
Keine Konvertierung in Codepages 867, 4899, 9048, 12712

9048
Keine Konvertierung in Codepages 424, 803, 856, 862, 916, 1255

Windows

Codepage:

1255
Keine Konvertierung in Codepages 867, 4899, 9048, 12712

5351
Keine Konvertierung in Codepage 803

Arabisch

Details zu CCSIDs und CCSID-Konvertierung für Arabisch.

In der folgenden Tabelle sind die nativen CCSIDs für Arabisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	420
AIX	1046, 1089

Plattform	Native CCSIDs
HP-UX	1089 (siehe Hinweis)
Windows	720, 864, 1256, 5352
HP Integrity NonStop Server, Solaris, Linux	1089
Anmerkung: Unter HP-UX wird nur der codierte ISO-Zeichensatz unterstützt. Der HP-UX-proprietäre codierte Zeichensatz arabic8 hat keine registrierte CCSID und wird nicht unterstützt.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

420
Keine Konvertierung in Codepage 5352

HP-UX, Solaris, Linux, HP Integrity NonStop Server, Tru64

Codepage:

1089
Keine Konvertierung in Codepage 720

Windows

Codepage:

720
Keine Konvertierung in Codepages 1089, 5352

5352
Keine Konvertierung in Codepage 720

Farsi

Details zu CCSIDs und CCSID-Konvertierung für Farsi.

In der folgenden Tabelle sind die nativen CCSIDs für Farsi auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1097
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1098 (siehe Hinweis)
Anmerkung: Die native CCSID für diese Plattformen wurde nicht standardisiert und ändert sich unter Umständen.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen.

Urdu

Details zu CCSIDs und CCSID-Konvertierung für Urdu.

In der folgenden Tabelle sind die nativen CCSIDs für Urdu auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	918
Windows	868
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1006

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

918
Keine Konvertierung in Codepage 1006

Thailändisch

Details zu CCSIDs und CCSID-Konvertierung für Thailändisch.

In der folgenden Tabelle sind die nativen CCSIDs für Thailändisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	838
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	874 (siehe Hinweis)
Anmerkung: Die native CCSID für diese Plattformen wurde nicht standardisiert und ändert sich unter Umständen.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen.

Laotisch

Details zu CCSIDs und CCSID-Konvertierung für Laotisch.

In der folgenden Tabelle sind die nativen CCSIDs für Laotisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1132
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1133

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen.

Vietnamesisch

Details zu CCSIDs und CCSID-Konvertierung für Vietnamesisch.

In der folgenden Tabelle sind die nativen CCSIDs für Vietnamesisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1130
Windows	1258, 5354

Plattform	Native CCSIDs
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1129

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

IBM i

Codepage:

1130
Keine Konvertierung in Codepages 1129, 5354

Japanisch mit lateinischem Einzelbytezeichensatz

Details zu CCSIDs und CCSID-Konvertierung für Japanisch mit lateinischem Einzelbytezeichensatz.

In der folgenden Tabelle sind die nativen CCSIDs für Japanisch mit lateinischem Einzelbytezeichensatz auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1027
AIX	932, 5050, 33722 (siehe Hinweis 1)
Windows	932, 943 (siehe Hinweis 2 und 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050
HP-UX	Nicht bekannt

Anmerkung:

1. 5050 und 33722 sind CCSIDs im Zusammenhang mit der Basiscodepage 954 unter AIX. Die vom Betriebssystem zurückgemeldete CCSID ist 33722.
2. Windows NT verwendete Codepage 932, diese wird jedoch am besten durch die CCSID 943 dargestellt. Jedoch unterstützen nicht alle Plattformen von WebSphere MQ diese CCSID.

   Unter WebSphere MQ for Windows wird die CCSID 932 zum Darstellen der Codepage 932 verwendet, jedoch kann eine Änderung an der Datei `../conv/table/ccsid.tbl` vorgenommen werden, wodurch die verwendete CCSID in 943 geändert wird.
3. WebSphere MQ unterstützt keine Codepages, die auf dem Standard JIS X 0213 (JIS2004) basieren.

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

1027
Keine Konvertierung in Codepages 932, 942, 943, 954, 5050, 33722

IBM i

Codepage:

1027
Keine Konvertierung in Codepage 932

AIX

Codepage:

932
Keine Konvertierung in Codepage 1027

5050
Keine Konvertierung in Codepage 1027

33722
Keine Konvertierung in Codepage 1027

Linux

Codepage:

943
Keine Konvertierung in Codepage 1027

5050
Keine Konvertierung in Codepage 1027

Solaris

Codepage:

943
Keine Konvertierung in Codepage 1027

5050
Keine Konvertierung in Codepage 1027

HP Integrity NonStop Server

Codepage:

943
Keine Konvertierung in Codepage 1027

5050
Keine Konvertierung in Codepage 1027

Japanisch, Katakana-Einzelbytezeichensatz

Details zu CCSIDs und CCSID-Konvertierung für Japanisch mit Katakana-Einzelbytezeichensatz.

In der folgenden Tabelle sind die nativen CCSIDs für Japanisch mit Katakana-Einzelbytezeichensatz auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	290
HP-UX	897
AIX	932, 5050, 33722 (siehe Hinweis 1)
Windows	932, 943 (siehe Hinweis 2 und 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Plattform	Native CCSIDs
Anmerkung: 1. 5050 und 33722 sind CCSIDs im Zusammenhang mit der Basiscodepage 954 unter AIX. Die vom Betriebssystem zurückgemeldete CCSID ist 33722. 2. Windows NT verwendete Codepage 932, diese wird jedoch am besten durch die CCSID 943 dargestellt. Jedoch unterstützen nicht alle Plattformen von WebSphere MQ diese CCSID. Unter WebSphere MQ for Windows wird die CCSID 932 zum Darstellen der Codepage 932 verwendet, jedoch kann eine Änderung an der Datei `../conv/table/ccsid.tbl` vorgenommen werden, wodurch die verwendete CCSID in 943 geändert wird. 3. WebSphere MQ unterstützt keine Codepages, die auf dem Standard JIS X 0213 (JIS2004) basieren. 4. Neben den obigen Konvertierungen unterstützen die WebSphere MQ -Produkte unter AIX, HP-UX, Solaris Linux und Tru64 die Konvertierung von CCSID 897 in die CCSIDs 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 und 1252.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

290
Keine Konvertierung in Codepages 932, 943, 954, 5050, 33722

IBM i

Codepage:

290
Keine Konvertierung in Codepage 932

AIX

Codepage:

932
Keine Konvertierung in Codepages 290, 897

5050
Keine Konvertierung in Codepages 290, 897

33722
Keine Konvertierung in Codepages 290, 897

HP-UX

Codepage:

897
Keine Konvertierung in Codepages 932, 943, 954, 5050, 33722

Linux

Codepage:

943
Keine Konvertierung in Codepages 290, 897

5050
Keine Konvertierung in Codepages 290, 897

Solaris

Codepage:

943
Keine Konvertierung in Codepages 290, 897

5050
Keine Konvertierung in Codepages 290, 897

HP Integrity NonStop Server

Codepage:

943
Keine Konvertierung in Codepages 290, 897

5050
Keine Konvertierung in Codepages 290, 897

Japanisch mit Mischung aus Kanji/Latein

Details zu CCSIDs und CCSID-Konvertierung für Japanisch mit Mischung aus Kanji/Latein.

In der folgenden Tabelle sind die nativen CCSIDs für Japanisch mit Mischung aus Kanji/Latein auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
IBM i, z/OS	1399, 5035 (siehe Hinweis 1)
AIX	932, 5050, 33722 (siehe Hinweis 2)
HP-UX	932, 954, 5039 (siehe Hinweis 3)
Windows	932, 943 (siehe Hinweis 4 und 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Anmerkung:

1. 5035 ist eine CCSID, die sich auf Codepage 939 bezieht
2. 5050 und 33722 sind CCSIDs im Zusammenhang mit der Basiscodepage 954 unter AIX. Die vom Betriebssystem zurückgemeldete CCSID ist 33722.
3. Die codierten Zeichensätze `japan15` und SJIS unter HP-UX werden durch CCSID 932 dargestellt. Diese verfügen über einige DBCS-Zeichen mit unterschiedlichen Darstellungen in SJIS, sodass 932 möglicherweise falsch konvertiert wird, wenn die Konvertierung nicht auf einem HP-UX -System ausgeführt wird. WebSphere MQ for HP-UX unterstützt 5039, die korrekte CCSID für HP SJIS. Es kann eine Änderung an der Datei `/var/mqm/conv/ccsid.tbl` vorgenommen werden, um die verwendete CCSID von 932 in 5039 zu ändern.
4. Windows NT verwendete Codepage 932, diese wird jedoch am besten durch die CCSID 943 dargestellt. Jedoch unterstützen nicht alle Plattformen von WebSphere MQ diese CCSID.

   Unter WebSphere MQ for Windows wird die CCSID 932 zum Darstellen der Codepage 932 verwendet, jedoch kann eine Änderung an der Datei `../conv/table/ccsid.tbl` vorgenommen werden, wodurch die verwendete CCSID in 943 geändert wird.
5. WebSphere MQ unterstützt keine Codepages, die auf dem Standard JIS X 0213 (JIS2004) basieren.

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

1399
Keine Konvertierung in Codepages 954, 5035, 5050, 33722

5035
Keine Konvertierung in Codepages 954, 1399, 5050, 33722

IBM i

Codepage:

1399
Keine Konvertierung in Codepage 5039

5035
Keine Konvertierung in Codepage 5039

HP-UX

Codepage:

932
Keine Konvertierung in Codepages 942, 943, 1399

954
Keine Konvertierung in Codepages 942, 943, 1399

5039
Keine Konvertierung in Codepages 942, 943, 1399

HP Integrity NonStop Server

Codepage:

943
Keine Konvertierung in Codepage 1399

5050
Keine Konvertierung in Codepage 1399

Japanisch, Kanji/Katakana gemischt

Details zu CCSIDs und CCSID-Konvertierung für Japanisch mit Mischung aus Kanji/Katakana.

Tabelle 582. Native CCSIDs für Japanisch mit Mischung aus Kanji/Katakana auf unterstützten Plattformen

Plattform	Native CCSIDs
z/OS	1390, 5026 (siehe Hinweis 1)
IBM i	5026 (siehe Hinweis 1)
AIX	932, 5050, 33722 (siehe Hinweis 2)
HP-UX	932, 954, 5039 (siehe Hinweis 3)
Windows	932, 943 (siehe Hinweis 4 und 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Tabelle 582. Native CCSIDs für Japanisch mit Mischung aus Kanji/Katakana auf unterstützten Plattformen (Forts.)

Plattform	Native CCSIDs
Anmerkung: 1. CCSID 1390 akzeptiert keine Zeichen in Kleinbuchstaben. 5026 ist eine CCSID, die sich auf Codepage 930 bezieht. CCSID 5026 ist die CCSID, die unter IBM i zurückgemeldet wird, wenn die Funktion für Japanisch Katakana (Doppelbytezeichensatz) ausgewählt ist. 2. 5050 und 33722 sind CCSIDs im Zusammenhang mit der Basiscodepage 954 unter AIX. Die vom Betriebssystem zurückgemeldete CCSID ist 33722. 3. Die codierten Zeichensätze `japan15` und SJIS unter HP-UX werden durch CCSID 932 dargestellt. Diese verfügen über einige DBCS-Zeichen mit unterschiedlichen Darstellungen in SJIS, sodass 932 möglicherweise falsch konvertiert wird, wenn die Konvertierung nicht auf einem HP-UX -System ausgeführt wird. WebSphere MQ for HP-UX unterstützt 5039, die korrekte CCSID für HP SJIS. Es kann eine Änderung an der Datei `/var/mqm/conv/ccsid.tbl` vorgenommen werden, um die verwendete CCSID von 932 in 5039 zu ändern. 4. Windows NT verwendete Codepage 932, diese wird jedoch am besten durch die CCSID 943 dargestellt. Jedoch unterstützen nicht alle Plattformen von WebSphere MQ diese CCSID. Unter WebSphere MQ for Windows wird CCSID 932 zum Darstellen der Codepage 932 verwendet, jedoch kann eine Änderung an der Datei `../conv/table/ccsid.tbl` vorgenommen werden, wodurch die verwendete CCSID in 943 geändert wird. 5. WebSphere MQ unterstützt keine Codepages, die auf dem Standard JIS X 0213 (JIS2004) basieren.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

1390
Keine Konvertierung in Codepages 954, 5026, 5050, 33722

Akzeptiert keine Zeichen in Kleinbuchstaben.

5026
Keine Konvertierung in Codepages 954, 1390, 5050, 33722

IBM i

Codepage:

5026
Keine Konvertierung in Codepages 1390, 5039

HP-UX

Codepage:

932
Keine Konvertierung in Codepages 942, 943, 1390

954
Keine Konvertierung in Codepages 942, 943, 1390

5039
Keine Konvertierung in Codepages 942, 943, 1390

HP Integrity NonStop Server

Codepage:

943
Keine Konvertierung in Codepage 1390

5050
Keine Konvertierung in Codepage 1390

Korean

Details zu CCSIDs und CCSID-Konvertierung für Koreanisch.

In der folgenden Tabelle sind die nativen CCSIDs für Koreanisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
z/OS, IBM i	933, 1364
AIX, HP-UX, Linux, HP Integrity NonStop Server, Solaris	970
Windows	949, 1363

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

933
Keine Konvertierung in Codepage 970

1364
Keine Konvertierung in Codepage 970

HP-UX

Codepage:

970
Keine Konvertierung in Codepages 949, 1363, 1364

Vereinfachtes Chinesisch

Details zu CCSIDs und CCSID-Konvertierung für vereinfachtes Chinesisch.

In der folgenden Tabelle sind die nativen CCSIDs für vereinfachtes Chinesisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
z/OS	935, 1388
IBM i	935, 1388
AIX	1383, 1386
HP-UX	1381 (siehe Hinweis 1)
Windows	1381, 1386 (siehe Hinweis 2)
Linux, HP Integrity NonStop Server, Solaris	1383

Plattform	Native CCSIDs
Anmerkung: 1. Die codierten Zeichensätze `prc15` und `hp15CN` unter HP-UX werden durch CCSID 1381 dargestellt. 2. Windows verwendet Codepage 936, diese wird jedoch am besten durch die CCSID 1386 dargestellt. Jedoch unterstützen nicht alle Plattformen von WebSphere MQ diese CCSID. Unter WebSphere MQ für Windows wird die CCSID 1381 zum Darstellen der Codepage 936 verwendet, jedoch kann eine Änderung an der Datei `../conv/table/ccsid.tbl` vorgenommen werden, wodurch die verwendete CCSID in 1386 geändert wird. 3. WebSphere MQ unterstützt Phase 1 des chinesischen GB18030-Standards. Unter z/OS, Linux, Windowsund Solaris wird Konvertierungsunterstützung zwischen Unicode (UTF-8 und UCS-2) und CCSID 1388 (EBCDIC mit GB18030 -Erweiterungen), Unicode (UTF-8 und UCS-2) und CCSID 5488 (GB18030 Phase 1) sowie zwischen CCSID 1388 und CCSID 5488 bereitgestellt. **Anmerkung:** Unter IBM i wird Unterstützung durch das Betriebssystem für die Konvertierung zwischen Unicode (UTF-8 und UCS-2) und CCSID 1388 (EBCDIC mit GB18030-Erweiterungen) bereitgestellt. Unter HP-UX ist derzeit auf dem Betriebssystem HP11 keine Unterstützung für GB18030 verfügbar. Unter HP11i stellt Patch PHCO_26456 Konvertierungsunterstützung zwischen GB18030 (CCSID 5488) und Unicode bereit. Für die Konvertierung zwischen GB18030 und 1388 (EBCDIC) wird keine Unterstützung bereitgestellt.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

935
Keine Konvertierung in Codepage 1383

1388
Keine Konvertierung in Codepage 1383

HP-UX

Codepage:

1381
Keine Konvertierung in Codepages 1383, 1386, 1388

Traditionelles Chinesisch

Details zu CCSIDs und CCSID-Konvertierung für traditionelles Chinesisch.

In der folgenden Tabelle sind die nativen CCSIDs für traditionelles Chinesisch auf unterstützten Plattformen dargestellt:

Plattform	Native CCSIDs
z/OS, IBM i	937
HP-UX	938, 950, 964 (siehe Hinweis)
Windows	950
AIX, HP Integrity NonStop Server, Solaris, Linux	950, 964

Plattform	Native CCSIDs
Anmerkung: Der codierte Zeichensatz `roc15` unter HP-UX wird durch CCSID 938 dargestellt.	

Alle Plattformen unterstützen die Konvertierung zwischen ihren nativen CCSIDs und den nativen CCSIDs anderer Plattformen mit folgenden Ausnahmen.

z/OS

Codepage:

937
Keine Konvertierung in Codepage 964

1388
Keine Konvertierung in Codepage 1383

HP-UX

Codepage:

938
Keine Konvertierung in Codepage 948

950
Keine Konvertierung in Codepage 948

964
Keine Konvertierung in Codepage 948

Linux, Solaris

Codepage:

964
Keine Konvertierung in Codepage 938

Unicode-Konvertierungsunterstützung

Einige Plattformen unterstützen die Konvertierung von Benutzerdaten in oder aus Unicode-Codierung. Die beiden unterstützen Formen der Unicode-Codierung sind UCS-2 (CCSIDs 1200, 13488 und 17584) und UTF-8 (CCSID 1208).

Der Begriff *UCS-2* wird häufig fälschlicherweise mit *UTF-16* gleichgesetzt. UCS-2 ist eine Codierung mit fester Breite, bei der jedes Zeichen zwei Bytes belegt. UTF-16 ist eine Codierung mit variabler Breite, bei der es sich um eine Obermenge von UCS-2 handelt. Zusätzlich zu den 2–Byte-Zeichen von UCS-2 enthält UTF-16 vier Bytes lange Zeichen, die als Ersatzzeichenpaar bezeichnet werden. WebSphere MQ unterstützt keine Ersatzzeichenpaare. Die Unterstützung für UTF-16 und UTF-8 in WebSphere MQ ist daher auf die Unicode-Zeichen beschränkt, die in UCS-2 codiert werden können.

Anmerkung: WebSphere MQ unterstützt keine Warteschlangenmanager-CCSIDs vom Typ UCS-2, sodass Nachrichtenheaderdaten nicht in UCS-2 codiert werden können.

WebSphere MQ AIX-Unterstützung für Unicode

Unter WebSphere MQ für AIX wird die Konvertierung in und aus Unicode-CCSIDs für die CCSIDs in der folgenden Tabelle unterstützt.

037	273	278	280	284	285
297	423	437	500	813	819
850	852	856	857	858	860

861	865	867	869	875	878
880	901	902	912	915	916
920	923	924	932	933	935
937	938	939	942	943	948
949	950	954	964	970	1026
1046	1089	1129	1130	1131	1132
1133	1140	1141	1142	1143	1144
1145	1146	1147	1148	1149	1200
1153	1156	1157	1208	1250	1251
1253	1254	1258	1280	1281	1282
1283	1284	1285	1363	1364	1381
1383	1386	1388	4899	5026	5035
5050	5346	5347	5348	5349	5350
5351	5352	5353	5354	5488	9044
9048	9449	12712	13488	17584	33722

WebSphere MQ HP-UX-Unterstützung für Unicode

Unter WebSphere MQ für HP-UX wird die Konvertierung in und aus Unicode-CCSIDs für die in der folgenden Tabelle aufgeführten CCSIDs unterstützt.

437	737	813	819	850	852
855	857	861	864	865	866
869	874	912	915	916	920
932	938	950	954	964	970
1051	1089	1140	1141	1142	1143
1144	1145	1146	1147	1148	1149
1200	1208	1250	1251	1252	1253
1254	1255	1256	1257	1258	1381
5050	5488	13488	33722		

WebSphere MQ für Windows, Solaris und Linux Unterstützung für Unicode

Unter WebSphere MQ für Windows, WebSphere MQ für Solaris und WebSphere MQ für Linux wird die Konvertierung in und aus Unicode-CCSIDs für die CCSIDs in der folgenden Tabelle unterstützt.

037	277	278	280	284	285
290	297	300	301	420	424
437	500	813	819	833	835
836	837	838	850	852	855
856	857	858	860	861	862
863	864	865	866	867	868
869	870	871	874	875	878

880	891	897	901	902	903
904	912	913 (5)	915	916	918
920	921	922	923	924	927
928	930	931 (1)	932 (2)	933	935
937	938 (3)	939	941	942	943
947	948	949	950	951	954 (4)
964	970	1006	1025	1026	1027
1040	1041	1.042	1043	1046	1047
1051	1088	1089	1097	1098	1112
1114	1115	1122	1123	1124	1129
1130	1132	1133	1140	1141	1142
1143	1144	1145	1146	1147	1148
1149	1153	1156	1157	1200	1208
1250	1251	1252	1253	1254	1255
1256	1257	1258	1275	1280	1281
1282	1283	1363	1364	1380	1381
1383	1386	1388	4899	5050	5346
5347	5348	5349	5350	5351	5352
5353	5354	5488 (5)	9044	9048	9449
12712	13488	17584	33722 (4)		

Anmerkungen:

1. 931 verwendet 939 für die Konvertierung.
2. 932 verwendet 942 für die Konvertierung.
3. 938 verwendet 948 für die Konvertierung.
4. 954 und 33722 verwenden 5050 für die Konvertierung.
5. Nur unter Windows, Linuxund Solaris.

IBM i-Unterstützung für Unicode

Details zur UNICODE-Unterstützung finden Sie in der entsprechenden IBM i -Veröffentlichung zu Ihrem Betriebssystem.

WebSphere MQ for z/OS-Unterstützung für Unicode

Unter WebSphere MQ for z/OS wird die Konvertierung in und aus Unicode-CCSIDs für die folgenden CCSIDs unterstützt:

273	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	1.042
1043	1046	1047	1051	1088	1089
1097	1098	1112	1114	1115	1122
1123	1124	1125	1126	1129	1130
1131	1132	1133	1137	1140	1141
1142	1143	1144	1145	1146	1147
1148	1149	1153	1154	1155	1156
1157	1158	1159	1160	1161	1162
1164	1200	1208	1250	1251	1252
1253	1254	1255	1256	1257	1258
1275	1276	1277	1280	1281	1282
1283	1284	1285	1351	1362	1363
1364	1370	1371	1380	1381	1385
1386	1388	1390	1399	4899	4909
4930	4933	4948	4951	4952	4960
4971	5012	5039	5104	5123	5142
5210	5346	5347	5348	5349	5350
5351	5352	5353	5354	5488	8482
8612	9027	9030	9044	9048	9049
9056	9061	9066	9238	9449	12712
13121	13218	13488	16684	16804	17248
17584	21427	28709			

Codierungsstandards auf 64-Bit-Plattformen

Dieser Abschnitt enthält Informationen zu Codierungsstandards auf 64-Bit-Plattformen und zu den bevorzugten Datentypen.

Bevorzugte Datentypen

Diese Typen ändern nie ihre Größe und sind sowohl auf 32-Bit- als auch auf 64-Bit-Plattformen von WebSphere MQ verfügbar:

Name	Länge
MQLONG	4 Byte
MQULONG	4 Byte
MQINT32	4 Byte
MQUINT32	4 Byte
MQINT64	8 Byte
MQUINT64	8 Byte

Standarddatentypen

Dieser Abschnitt enthält Informationen zu den Standarddatentypen in 32-Bit-Anwendungen unter UNIX, in 64-Bit-Anwendungen unter UNIX und 64-Bit-Anwendungen unter Windows.

32-Bit-Anwendungen unter UNIX

Dieser Abschnitt wurde zu Vergleichszwecken aufgenommen. Die darin enthaltenen Angaben basieren auf Solaris. Eventuelle Unterschiede zu anderen UNIX-Plattformen werden in einem entsprechenden Hinweis genannt:

Name	Länge
char	1 Byte
short	2 Bytes
Int	4 Byte
long	4 Byte
float	4 Byte
double	8 Byte
long double	16 Bytes Beachten Sie, dass unter AIX und Linux PPC ein langer Doppelbytezeichen 8 Byte beträgt.
pointer	4 Byte
ptrdiff_t	4 Byte
size_t	4 Byte
time_t	4 Byte
clock_t	4 Byte
wchar_t	4 Byte Hinweis: Unter AIX hat der Datentyp "wchar_t" eine Länge von zwei Bytes.

64-Bit-Anwendungen unter UNIX

Die Angaben in diesem Abschnitt basieren auf Solaris. Eventuelle Unterschiede zu anderen UNIX-Plattformen werden in einem entsprechenden Hinweis genannt:

Name	Länge
char	1 Byte
short	2 Bytes
Int	4 Byte
long	8 Byte
float	4 Byte
double	8 Byte
long double	16 Bytes
	Beachten Sie, dass unter AIX und Linux PPC ein langer Doppelbytezeichen 8 Byte beträgt.
pointer	8 Byte
ptrdiff_t	8 Byte
size_t	8 Byte
time_t	8 Byte
clock_t	8 Byte
	Hinweis: Auf den anderen UNIX-Plattformen hat der Datentyp "clock_t" eine Länge von vier Bytes.
wchar_t	4 Byte
	Hinweis: Unter AIX hat der Datentyp "wchar_t" eine Länge von zwei Bytes.

64-Bit-Anwendungen unter Windows

Name	Länge
char	1 Byte
short	2 Bytes
Int	4 Byte
long	4 Byte
float	4 Byte
double	8 Byte
long double	8 Byte
pointer	8 Byte
	Beachten Sie, dass alle pointer-Datentypen eine Länge von acht Bytes haben.
ptrdiff_t	8 Byte
size_t	8 Byte
time_t	8 Byte
clock_t	4 Byte

Name	Länge
wchar_t	2 Bytes
WORD	2 Bytes
DWORD	4 Byte
HANDLE	8 Byte
HFILE	4 Byte

Codierungsaspekte unter Windows

HANDLE hf;

Verwenden Sie

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

Verwenden Sie nicht

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

Dieser Code würde zu einem Fehler führen.

size_t len fgets

Verwenden Sie

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

Verwenden Sie nicht

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf

Verwenden Sie

```
printf("My struc pointer: %p", pMyStruc);
```

Verwenden Sie nicht

```
printf("My struc pointer: %x", pMyStruc);
```

Wenn Sie eine hexadezimale Ausgabe benötigen, müssen die oberen und unteren vier Bytes gesondert ausgegeben werden.

char *ptr

Verwenden Sie

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

Verwenden Sie nicht

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Verwenden Sie

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

Verwenden Sie nicht

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

Verwenden Sie

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

Verwenden Sie nicht

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

Verwenden Sie

```
MQLONG SBCSprt;

sscanf(line, "%d", &SBCSprt);
```

Verwenden Sie nicht

```
MQLONG SBCSprt;

sscanf(line, "%1d", &SBCSprt);
```

%ld versucht, einen 8-Byte-Typ in einen 4-Byte-Typ zu ändern; verwenden Sie %l also nur, wenn es sich in Ihrem Fall tatsächlich um den Datentyp long handelt. MQLONG, UINT32 und INT32 sind

ebenso wie der Datentyp `int` auf allen WebSphere MQ-Plattformen als Datentypen mit einer Länge von vier Bytes definiert:

SOAP-Referenz

Die Referenzinformationen zu WebSphere MQ Transport for SOAP sind alphabetisch angeordnet.

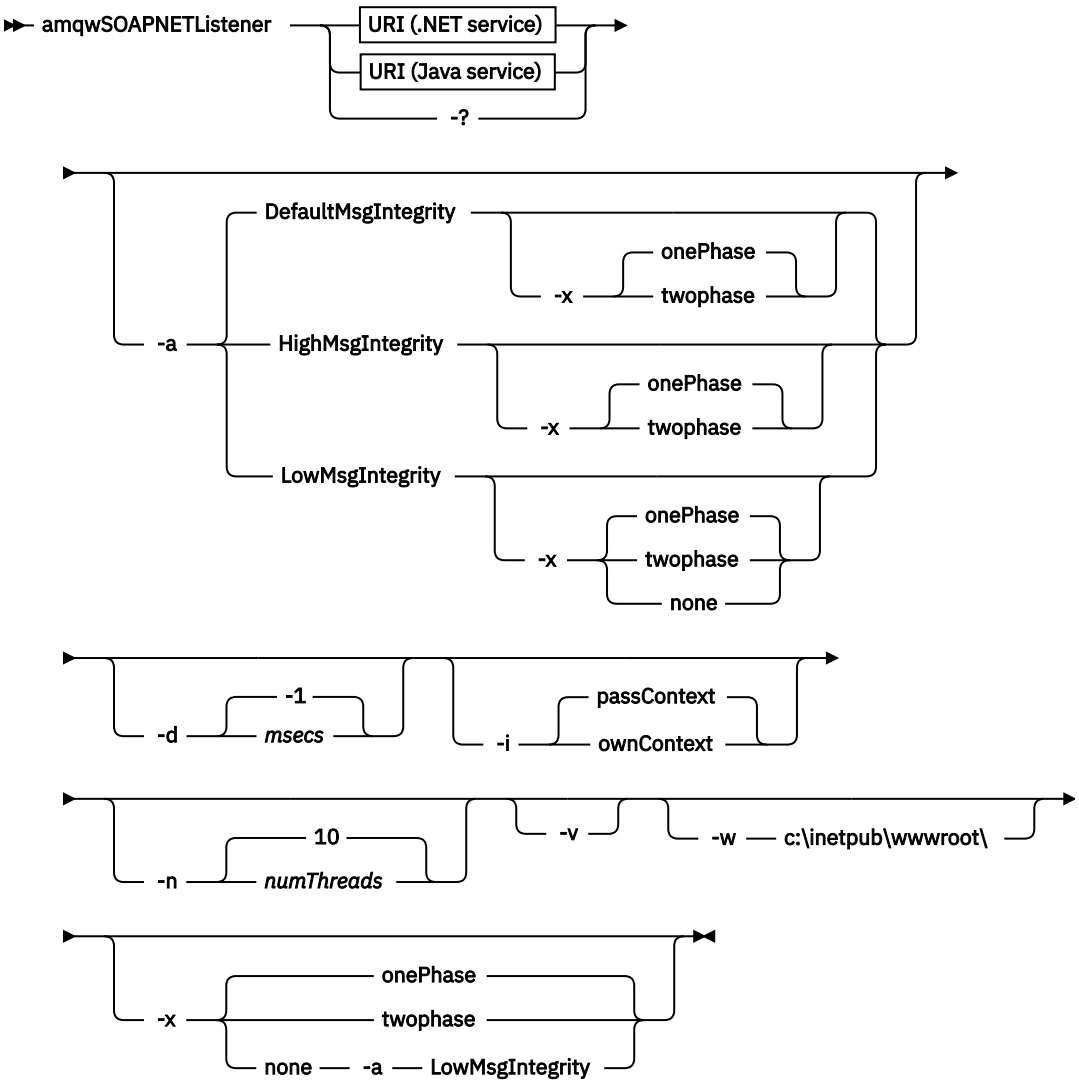
amqwSOAPNETListener: SOAP-Listener von IBM WebSphere MQ für .NET Framework 1 oder 2

Syntax und Parameter des SOAP-Empfangsprogramms von WebSphere MQ für .NET Framework 1 oder 2.

Verwendungszweck

Startet den SOAP-Listener von IBM WebSphere MQ für .NET Framework 1 oder 2.

.NET



Erforderliche Parameter

URI ***platform***

Siehe „URI-Syntax und -Parameter für die Web-Service-Implementierung” auf Seite 1016.

-?
Hilfetext ausgeben, der beschreibt, wie der Befehl verwendet wird.

Optionale Parameter

-a ***integrityOption***
integrityOption gibt das Verhalten von SOAP-Empfangsprogrammen von WebSphere MQ an, wenn es nicht möglich ist, eine fehlgeschlagene Anforderungsnachricht in die Warteschlange für nicht zustellbare Nachrichten einzureihen. *integrityOption* kann einen der folgenden Werte haben:

DefaultMsgIntegrity
Bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung mit der ursprünglichen, verworfenen Nachricht fort. Bei persistenten Nachrichten zeigt es eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet. DefaultMsgIntegrity gilt, wenn die Option -a ausgelassen wird oder wenn *integrityOption* nicht angegeben ist.

LowMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung fort, wobei die Nachricht verworfen wird.

HighMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Wenn -x none angegeben ist, muss -a LowMsgIntegrity angegeben sein. Wenn die Flags nicht kompatibel sind, wird das Implementierungsdienstprogramm mit einer Fehlernachricht beendet, ohne dass irgendein Implementierungsschritt ausgeführt wurde.

-d ***msecs***
msecs gibt an, wie lange (in Millisekunden) das SOAP-Empfangsprogramm von WebSphere MQ aktiv bleibt, wenn in einem beliebigen Thread Nachrichten eingegangen sind. Wenn für *msecs* der Wert -1 festgelegt ist, bleibt das Empfangsprogramm für unbegrenzte Zeit aktiv.

-i ***Context***
Context gibt an, ob die Empfangsprogramme den Identitätskontext übergeben. *Context* kann die folgenden Werte haben:

passContext
Identitätskontext der ursprünglichen Anforderungsnachricht in die Antwortnachricht einfügen. Das SOAP-Empfangsprogramm prüft, ob es berechtigt ist, den Kontext aus der Anforderungswarteschlange zu speichern und an die Antwortwarteschlange zu übergeben. Es führt diese Prüfungen zur Laufzeit aus, wenn es die Anforderungswarteschlange zum Speichern des Kontextes und die Antwortwarteschlange zum Übergeben des Kontextes öffnet. Wenn es nicht über die erforderliche Berechtigung verfügt oder der MQOPEN-Aufruf fehlschlägt, wird die Antwortnachricht nicht verarbeitet. Die Antwortnachricht wird in die Warteschlange für nicht zustellbare Nachrichten eingereiht, wobei der Header der nicht zustellbaren Nachricht den Rückkehrcode aus dem fehlgeschlagenen MQOPEN-Aufruf enthält. Das Empfangsprogramm fährt anschließend mit der normalen Verarbeitung der nachfolgenden eingehenden Nachrichten fort.

ownContext
Das SOAP-Empfangsprogramm übergibt keinen Kontext. Der zurückgegebene Kontext gibt die Benutzer-ID an, unter der das Empfangsprogramm ausgeführt wird, und nicht die Benutzer-ID, die die ursprüngliche Anforderungsnachricht erstellt hat.

Die Felder im Ursprungskontext werden vom Warteschlangenmanager festgelegt, nicht vom SOAP-Empfangsprogramm.

-n ***numThreads***
numThreads gibt die Anzahl der Threads in den generierten Startscripts für das SOAP-Empfangsprogramm von WebSphere MQ an. Der Standardwert ist 10. Es kann sinnvoll sein, diesen Wert zu erhöhen, wenn der Nachrichtendurchsatz hoch ist.

-v
-v legt die ausführliche Ausgabe von externen Befehlen fest. Fehlernachrichten werden immer angezeigt. Verwenden Sie -v, um Befehle auszugeben, die Sie anpassen können, um angepasste Implementierungsscripts zu erstellen.

-w ***serviceDirectory***
serviceDirectory ist das Verzeichnis, das den Web-Service enthält.

-x ***transactionality***
transactionality gibt den Typ der Transaktionssteuerung für das Empfangsprogramm an. Für *transactionality* kann einer der folgenden Werte festgelegt sein:

onePhase
Es wird die Einphasenunterstützung von IBM WebSphere MQ verwendet. Wenn das System während der Verarbeitung ausfällt, wird die Anforderungsnachricht erneut der Anwendung zugestellt. WebSphere MQ -Transaktionen stellen sicher, dass die Antwortnachrichten genau einmal geschrieben werden.

twoPhase
Die Zweiphasenunterstützung wird verwendet. Wenn der Service entsprechend geschrieben wurde, wird die Nachricht genau einmal zugestellt, in Koordination mit anderen Ressourcen und innerhalb einer einzelnen festgeschriebenen Ausführung des Service. Diese Option gilt nur für Verbindungen mit Serverbindung.

none
Keine Transaktionsunterstützung. Wenn das System während der Verarbeitung ausfällt, kann die Anforderungsnachricht verloren gehen, auch wenn sie persistent ist. In diesem Fall ist es möglich, dass der Service ausgeführt wurde oder nicht und dass Antwort-, Berichts- oder nicht zustellbare Nachrichten geschrieben wurden oder nicht.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Weitere Informationen finden Sie in der Beschreibung zum Flag -a.

Beispiel für .NET

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

amqswsdl: WSDL für .NET Framework 1- oder 2-Service generieren

amqswsdl generiert die WSDL für die Klasse eines Web-Service, der für .NET Framework 1 oder 2 geschrieben wurde. Dabei wird der von Ihnen für WebSphere MQ Transport for SOAP angegebene URI in die generierte WSDL eingefügt.

Verwendungszweck

Verwenden Sie **amqswsdl**, um die WSDL mit dem URI eines für WebSphere MQ implementierten Service zu generieren. Mit dieser WSDL können Sie Client-Proxys generieren.

Parameter

***escapedUri* (Eingabe)**
Der URI des Service, wobei allen "&" wie folgt Escapezeichen nachgestellt werden müssen: "&amp.". Beispiel:

```
"jms:/queue?destination=REQUESTDOTNET
&amp.initialContextFactory=com.ibm.mq.jms.Nojndi
```

```
&amp.connectionFactory=(connectQueueManager(QM1)binding(server))
&amp.targetService=Quote.asmx"
```

***className*.asmx (Eingabe)**
Die Serviceklasse.

***className*.wsdl (Ausgabe)**
Die WSDL des Service.

Beschreibung

Wenn die Klasse mit dem Code-Behind-Programmiermodell implementiert wird, müssen Sie die Datei *className*.dll erstellen und in ./bin ablegen.

amqwclientconfig: Implementierungsdeskriptor des Web-Service-Clients Axis 1.4 für WebSphere MQ Transport for SOAP erstellen

amqwclientconfig erstellt die Implementierungsdeskriptordatei client-config.wsdd des Axis 1.4-Clients.

Verwendungszweck

Der Befehl fügt den Transport jms:/ zum Deskriptor hinzu und registriert java:com.ibm.mq.soap.transport.jms.WMQSender als Klasse zur Bearbeitung von SOAP-Anforderungen für den Transport jms:.

Syntax

Beschreibung

amqwclientconfig ruft **amqwsetcp** auf, um den CLASSPATH festzulegen, und führt den folgenden Befehl aus:

```
java org.apache.axis.utils.Admin client "%WMQSOAP_HOME%\bin\amqwclientTransport.wsdd"
```

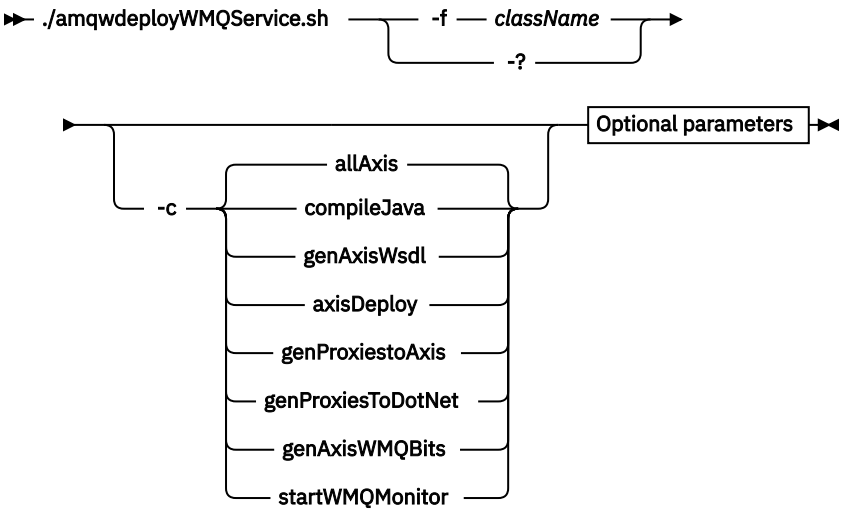
amqwdeployWMQService: Implementierungsdienstprogramm für Web-Services

Das Implementierungsdienstprogramm bereitet eine Serviceklasse auf die Verwendung als Web-Service vor, wobei WebSphere MQ als Transportprotokoll verwendet wird.

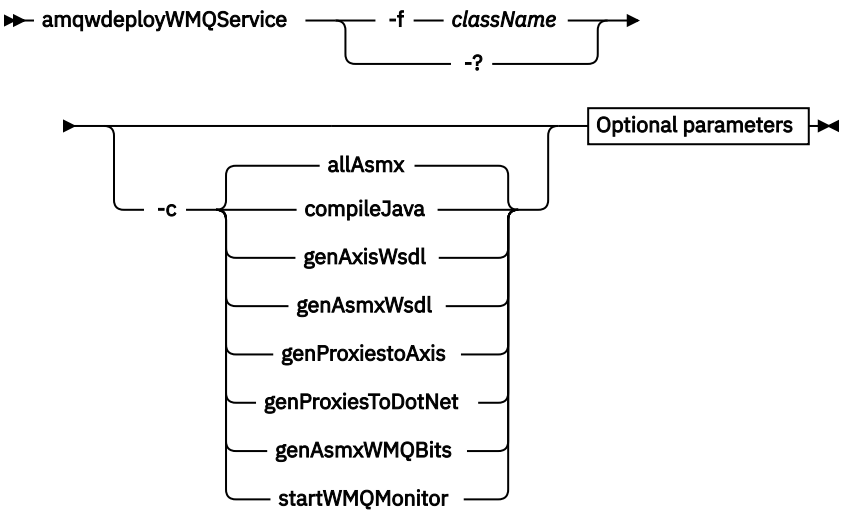
Verwendungszweck

Verwenden Sie das Implementierungsdienstprogramm, um die benötigten Dateien zum Implementieren eines Axis 1.4-, .NET Framework 1- oder .NET Framework 2-Service zu generieren. Verwenden Sie die Dateien, um einen von IBM WebSphere MQ aufgerufenen Service zu implementieren. Die von **amqwdeployWMQService** generierten Dateien werden in „Ausgabedateien von amqwdeployWMQService” auf Seite 985angezeigt.

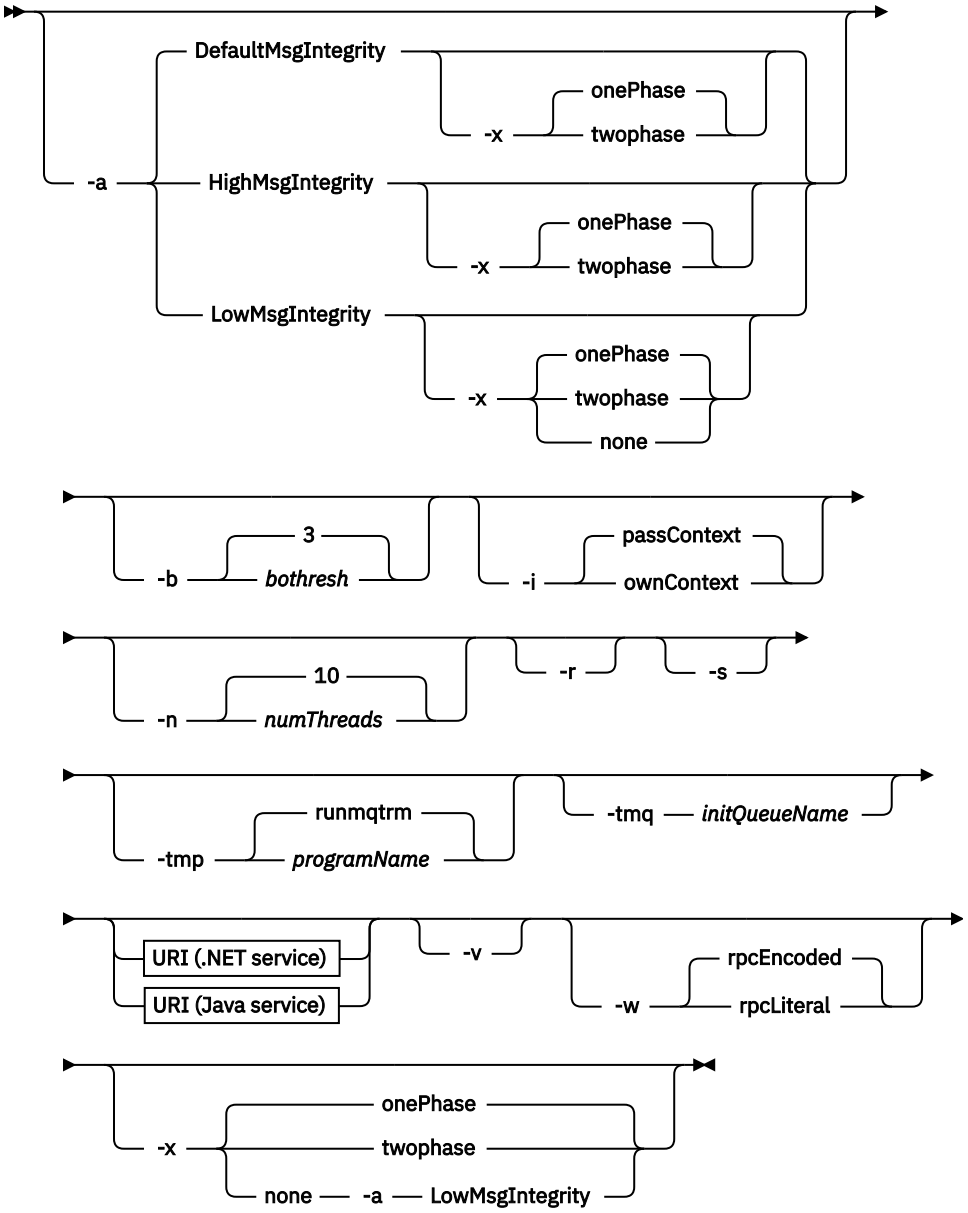
Syntax diagram
UNIX and Linux systems



Windows



Optional parameters



Erforderliche Parameter

-f ***className***

className ist der Name der zu implementierenden Klasse. Für Axis-Services ist *className* die Java-Quellendatei und für .NET-Services die Datei `.asmx` . In Abbildung 11 auf Seite 981 ist die Implementierung eines Axis-Service dargestellt, in Abbildung 12 auf Seite 981 die eines .NET-Service.

```
amqwdeployWMQService -f javaDemos/service/StockQuoteAxis.java
```

Abbildung 11. Beispiel: Implementierung eines Axis-Service

```
amqwdeployWMQService -f StockQuoteDotNet.asmx
```

Abbildung 12. Beispiel: Implementierung eines .NET-Service

Für Java muss *className* vollständig durch den Paketnamen qualifiziert sein. Er kann als Pfadname mit Verzeichnistrennzeichen oder als Klassenname mit Punkten als Trennzeichen angegeben werden. Die generierte Klasse befindet sich in `./generated/client/remote/`*`path name`*. Obwohl das

Verzeichnis für einen .NET-Service angegeben werden kann, befinden sich generierte Java-Proxys immer im Verzeichnis `./generated/client/remote/dotNetService`.

Wenn Sie einen URI mit der Option -u angeben und innerhalb des URI *targetService* angeben, prüft das Implementierungsdienstprogramm den *className*. *className* muss mit *targetService* übereinstimmen. Wenn Klasse und Service nicht übereinstimmen, zeigt das Implementierungsdienstprogramm eine Fehlernachricht an und wird beendet.

-?
Hilfetext ausgeben, der beschreibt, wie der Befehl verwendet wird.

Optionale Parameter

-a ***integrityOption***
integrityOption gibt das Verhalten von SOAP-Empfangsprogrammen von WebSphere MQ an, wenn es nicht möglich ist, eine fehlgeschlagene Anforderungsnachricht in die Warteschlange für nicht zustellbare Nachrichten einzureihen. *integrityOption* kann einen der folgenden Werte haben:

DefaultMsgIntegrity
Bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung mit der ursprünglichen, verworfenen Nachricht fort. Bei persistenten Nachrichten zeigt es eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet. DefaultMsgIntegrity gilt, wenn die Option -a ausgelassen wird oder wenn *integrityOption* nicht angegeben ist.

LowMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung fort, wobei die Nachricht verworfen wird.

HighMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Wenn -x none angegeben ist, muss -a LowMsgIntegrity angegeben sein. Wenn die Flags nicht kompatibel sind, wird das Implementierungsdienstprogramm mit einer Fehlernachricht beendet, ohne dass irgendein Implementierungsschritt ausgeführt wurde.

-b ***bothresh***
bothresh gibt die Einstellung für den Rücksetzschwellenwert an, die für die Anforderungswarteschlange gelten soll. Der Standardwert ist 3.

-c ***operation***
operation gibt an, welcher Teil des Implementierungsprozesses auszuführen ist. *operation* ist eine der folgenden Optionen:

allAxis
Alle Kompilier-und Konfigurationsschritte für einen Axis-oder Java-Service ausführen[4].

compileJava
Kompilieren Sie den Java-Service: `.java` in `.class`.

genAxisWsdl
WSDL generieren: `.class` in `.wsdl`.

axisDeploy
Klassendatei implementieren: `.wsdl` in `.wsdd`, `.wsdd` anwenden.

genProxiestoAxis
Proxys generieren: `.wsdl` in `.java` und `.class`.

[4] Standardwert, wenn *className* die Erweiterung .java hat

genAxisWMQBits
IBM WebSphere MQ-Warteschlangen, SOAP-Empfangsprogramme von IBM WebSphere MQ und Auslöser für einen Axis-Service einrichten.

allAsmx
Alle Konfigurationsschritte für einen .NET-Service ausführen[5].

genAsmxWsdl
WSDL generieren: `.asmx` in `.wsdl`.

genProxiesToDotNet
Proxys generieren: `.wsdl` in `.java`, `.class`, `.cs` und `.vb`.

genAsmxWMQBits
IBM WebSphere MQ-Warteschlangen, SOAP-Empfangsprogramme von IBM WebSphere MQ und Auslöser einrichten.

startWMQMonitor
Auslösemonitor für WebSphere MQ-SOAP-Services starten.

Anmerkung: **runmqtrm** wird unter der Benutzer-ID `mqm` ausgeführt. Wenn die Sicherheit kritisch ist, müssen Sie sicherstellen, dass die Empfangsprogramme unter entsprechenden Benutzer-IDs gestartet werden.

-i ***Context***
Context gibt an, ob die Empfangsprogramme den Identitätskontext übergeben. *Context* kann die folgenden Werte haben:

passContext
Identitätskontext der ursprünglichen Anforderungsnachricht in die Antwortnachricht einfügen. Das SOAP-Empfangsprogramm prüft, ob es berechtigt ist, den Kontext aus der Anforderungswarteschlange zu speichern und an die Antwortwarteschlange zu übergeben. Es führt diese Prüfungen zur Laufzeit aus, wenn es die Anforderungswarteschlange zum Speichern des Kontextes und die Antwortwarteschlange zum Übergeben des Kontextes öffnet. Wenn es nicht über die erforderliche Berechtigung verfügt oder der MQOPEN-Aufruf fehlschlägt, wird die Antwortnachricht nicht verarbeitet. Die Antwortnachricht wird in die Warteschlange für nicht zustellbare Nachrichten eingereiht, wobei der Header der nicht zustellbaren Nachricht den Rückkehrcode aus dem fehlgeschlagenen MQOPEN-Aufruf enthält. Das Empfangsprogramm fährt anschließend mit der normalen Verarbeitung der nachfolgenden eingehenden Nachrichten fort.

ownContext
Das SOAP-Empfangsprogramm übergibt keinen Kontext. Der zurückgegebene Kontext gibt die Benutzer-ID an, unter der das Empfangsprogramm ausgeführt wird, und nicht die Benutzer-ID, die die ursprüngliche Anforderungsnachricht erstellt hat.

Die Felder im Ursprungskontext werden vom Warteschlangenmanager festgelegt, nicht vom SOAP-Empfangsprogramm.

-n ***numThreads***
numThreads gibt die Anzahl der Threads in den generierten Startscripts für das SOAP-Empfangsprogramm von WebSphere MQ an. Der Standardwert ist 10. Es kann sinnvoll sein, diesen Wert zu erhöhen, wenn der Nachrichtendurchsatz hoch ist.

-r
-r gibt an, dass alle eventuell vorhandenen Definitionen von Anforderungs- oder Auslösemonitorwarteschlangen ersetzt werden. Auslösemonitorwarteschlangen werden nur ersetzt, wenn auch -tmq angegeben ist. Warteschlangen werden mit Standardattributen erneut erstellt und vorhandene Nachrichten in den Warteschlangen werden gelöscht. Wenn die Option -r nicht verwendet wird, werden vorhandene Warteschlangendefinitionen nicht geändert und vorhandene Nachrichten werden nicht gelöscht. Indem Sie -r nicht angeben, stellen Sie sicher, dass alle angepassten Warteschlangenattribute erhalten bleiben.

[5] Standardwert, wenn *className* die Erweiterung .asmx hat.

-s
Empfangsprogramm so konfigurieren, dass es als WebSphere MQ-Service ausgeführt wird. Wenn sowohl -s als auch -tmq angegeben sind, zeigt das Implementierungsdienstprogramm eine Fehlernachricht an und wird beendet.

-tmp ***programName***
programName gibt den Namen eines Auslösemonitorprogramms an. Die Verwendung von `-tmp` *programName* ist in einer UNIX- oder Linux-Umgebung als Alternative zu **runmqtrm** möglich. Programme, die damit ausgelöst werden, werden mit der Berechtigung `mqm` ausgeführt.

Beispiel:

```
amqwdeployWMQService -f javaDemos/service/StockQuoteAxis.java
-tmq trigger.monitor.queue -tmp trigmon
```

-tmq ***queueName***
queueName gibt den Namen einer Auslösemonitorwarteschlange an. IBM WebSphere MQ-Prozessdefinitionen werden erstellt, um das automatische Auslösen von SOAR-Listenern von WebSphere MQ mit dem zugehörigen Namen der Auslösemonitorwarteschlange zu konfigurieren. Wenn die Option nicht angegeben ist, definiert das Implementierungsdienstprogramm keine Auslösekonfiguration. Wenn sowohl -s als auch -tmq angegeben sind, zeigt das Implementierungsdienstprogramm eine Fehlernachricht an und wird beendet.

URI ***platform***
Siehe „URI-Syntax und -Parameter für die Web-Service-Implementierung” auf Seite 1016.

-v
-v legt die ausführliche Ausgabe von externen Befehlen fest. Fehlernachrichten werden immer angezeigt. Verwenden Sie -v, um Befehle auszugeben, die Sie anpassen können, um angepasste Implementierungsscripts zu erstellen.

-w
-w bestimmt, welche Art von WSDL generiert wird. Die Standardeinstellung ist rpcEncloded, aus Gründen der Kompatibilität mit früheren Releases von WebSphere MQ Transport for SOAP. Verwenden Sie rpcLiteral, um eine WSDL zu erstellen, die mit der Client-Proxy-Generierung bei Axis 2 kompatibel ist. rpcEncoded ist nicht mit den WS-I-Empfehlungen kompatibel.

-x ***transactionality***
transactionality gibt den Typ der Transaktionssteuerung für das Empfangsprogramm an. Für *transactionality* kann einer der folgenden Werte festgelegt sein:

onePhase
Es wird die Einphasenunterstützung von IBM WebSphere MQ verwendet. Wenn das System während der Verarbeitung ausfällt, wird die Anforderungsnachricht erneut der Anwendung zugestellt. WebSphere MQ -Transaktionen stellen sicher, dass die Antwortnachrichten genau einmal geschrieben werden.

twoPhase
Die Zweiphasenunterstützung wird verwendet. Wenn der Service entsprechend geschrieben wurde, wird die Nachricht genau einmal zugestellt, in Koordination mit anderen Ressourcen und innerhalb einer einzelnen festgeschriebenen Ausführung des Service. Diese Option gilt nur für Verbindungen mit Serverbindung.

none
Keine Transaktionsunterstützung. Wenn das System während der Verarbeitung ausfällt, kann die Anforderungsnachricht verloren gehen, auch wenn sie persistent ist. In diesem Fall ist es möglich, dass der Service ausgeführt wurde oder nicht und dass Antwort-, Berichts- oder nicht zustellbare Nachrichten geschrieben wurden oder nicht.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Weitere Informationen finden Sie in der Beschreibung zum Flag -a.

Fehler

Wenn unter Windows Fehler bei **amqswsdl** gemeldet werden, können diese möglicherweise behoben werden, indem Sie den folgenden Befehl absetzen, um `.asmx`-Dateien als Services zu registrieren.

```
%windir%/Microsoft.NET/Framework/version number/aspnet_regiis.exe -ir
```

Der Fehler kommt meist bei Systemen vor, auf denen IIS nicht installiert ist oder nach NET installiert wurde. Der Fehler tritt auf, wenn **amqswsdl** die `.wsdl`-Dateien generiert.

Anmerkung: Außerdem müssen die Registrierungsschlüssel zulassen, dass das Empfangsprogramm die Services aufruft. Wenn Sie eigene, angepasste Implementierungsprozeduren verwenden, tritt der Fehler möglicherweise erst während der Laufzeit auf.

Ausgabedateien von amqwdeployWMQService

In diesem Abschnitt finden Sie eine Liste der Verzeichnisse und Dateien, die von **amqwdeployWMQService** ausgegeben werden.

*Tabelle 583. Ausgabedateien von **amqwdeployWMQService***

Ausgaben	Beschreibung	Ausgabeverzeichnis	Dateiname
`.class`	Kompilierte Java-Quellendatei	`./generated/server/`*server package*	*classname*`.class`
`.wsdl`	Servicebeschreibung	`./generated`	*className*`Axis_Wmq.wsdl` *className*`DotNet_Wmq.wsdl`
`.wsdd`	Client- und Serviceimplementierungsdateien von Axis	`./`	`client-config.wsdd` `server-config.wsdd`
		`./generated/server/`*server package*	*className*`_deploy.wsdd` *className*`_undeploy.wsdd`
Quellcode des Clients (`.vb`, `.cs`, `.java`)	.NET-Client-Stubs für Axis-Service	`./generated/client`	*classname*`AxisService.cs` *classname*`AxisService.vb`
	.NET-Client-Stubs für .NET-Service	`./generated/client`	*classname*`DotNet.cs` *classname*`DotNet.vb`

*Tabelle 583. Ausgabedateien von **amqwdeployWMQService** (Forts.)*			
Ausgaben	**Beschreibung**	**Ausgabeverzeichnis**	**Dateiname**
Client-Hilfeprogramm (.java and .class)	Java-Client-Proxys für .NET-Service	./generated/server/soap/client/remote/dotnetService	*className*DotNet.class *className*DotNet.java *className*DotNetLocator.class *className*DotNetLocator.java *className*DotNetSoap12Stub.class *className*DotNetSoap12Stub.java *className*DotNetSoap_BindingStub.class *className*DotNetSoap_BindingStub.java *className*DotNetSoap_PortType.class *className*DotNetSoap_PortType.java
	Java-Client-Proxys für Axis-Service	./generated/server/soap/client/remote/*client package*	SoapServer*className*AxisBindingSoapStub.class SoapServer*className*AxisBindingSoapStub.java *className*Axis.class *className*Axis.java *className*AxisService.class *className*AxisService.java *className*AxisServiceLocator.class *className*AxisServiceLocator.java
Scripts (.cmd und .sh)	Empfangsprogrammscripts	/generated/server	startWMQJListener.cmd startWMQJListener.sh startWMQNListener.cmd endWMQJListener.cmd endWMQJListener.sh endWMQNListener.cmd

Hinweise zur Verwendung von amqwdeployWMQService

In diesem Abschnitt werden die von **amqwdeployWMQService** ausgeführten Tasks beschrieben.

Das Implementierungsdienstprogramm führt die folgenden Aktionen aus.

1. Prüft die Pfade zu den folgenden Dateien:
   - axis.jar.
   - *WMQSOAP_HOME*/java/lib/com.ibm.mq.soap.jar.
   - Unter Windows: csc.exe
2. Unter Windows verwendet es entweder %SystemRoot%\Microsoft.NET\Framework\v1.1.432 oder, wenn der C#-Compiler installiert ist, den Pfad zu csc.exe als Pfad zu .NET Framework.

   Anmerkung: Wenn bei Ihnen Microsoft Visual Studio 2008 (Version 9) installiert ist, befindet sich wsdl.exe nicht im Pfad zu csc.exe. Sie müssen den Pfad zu.NET-Framework zu Ihrer Pfadvariablen hinzufügen, zum Beispiel:

   ```
   Set Path=C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;%Path%
   ```
3. Erstellt das Verzeichnis ./generated sowie die erforderlichen Unterverzeichnisse, falls es diese noch nicht gibt.
4. Für Java-Services wird die Quelle in *className*.class kompiliert.

5. Erstellt die WSDL.
6. Für Java-Services werden Implementierungsdeskriptordateien *className*_deploy.wsdd und *className*_undeploy.wsdd erstellt.
7. Für Java-Services: Erstellt oder aktualisiert die Axis-Implementierungsdeskriptordatei server-config.wsdd.
8. Generiert die Client-Proxys für Java, C# und Visual Basic aus der WSDL.

   Anmerkung: Unter Windows generiert das Implementierungsdienstprogramm Proxys für Visual Basic und C# ungeachtet der Sprache, in der der Service geschrieben ist. Die WSDL und die daraus generierten Proxys enthalten den entsprechenden URI zum Aufrufen des Service.

   a.
   ```
   jms:/queue?destination=SOAPN.demos@WMQSOAP.DEMO.QM
   &connectionFactory=(connectQueueManager(WMQSOAP.DEMO.QM))
   &initialContextFactory=com.ibm.mq.jms.Nojndi
   &targetService=StockQuoteDotNet.asmx
   &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
   ```

   Abbildung 13. Beispiel-URI in generiertem .NET-Client zum Aufrufen eines .NET-Service

   b.
   ```
   jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
   &connectionFactory=(connectQueueManager(WMQSOAP.DEMO.QM))
   &initialContextFactory=com.ibm.mq.jms.Nojndi
   &targetService=soap.server.StockQuoteAxis.java
   &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
   ```

   Abbildung 14. Beispiel-URI in generiertem .NET-Client zum Aufrufen eines Axis 1-Service
9. Kompiliert die Java-Proxys.
10. Erstellt eine WebSphere MQ-Warteschlange, *requestQueue*, in die die Anforderungen für den Service eingereiht werden. Der Standardwarteschlangenname hat das Format SOAPJ.*directory*oder Sie können *requestQueue* in der URI-Option -u angeben.
11. Erstellt Befehls- und Shell-Script-Dateien zum Starten der SOAP-Empfangsprogramme von WebSphere MQ, die die Anforderungswarteschlange verarbeiten.
12. Wenn die Option -tmq verwendet wurde, erstellt das Implementierungsdienstprogramm WebSphere MQ-Definitionen, um Prozesse des SOAP-Empfangsprogramms von WebSphere MQ automatisch auszulösen.

    - Das Implementierungsdienstprogramm verwendet das Attribut APPLICID des Befehls **runmqsc** DEFINE PROCESS zum Einfügen eines Befehls, der das Empfangsprogramm startet. Der Befehl hat den Namen des darin eingebetteten Implementierungsverzeichnisses. Das Feld APPLICID hat eine maximale Länge von 256 Zeichen, was die maximale Länge des Implementierungsverzeichnisses begrenzt. Die Verzeichnisbegrenzung für Java-Services lautet wie folgt:
      - UNIX and Linux-Systeme: 218
      - Windows: 197 abzüglich der Länge des Namens der Anforderungswarteschlange.

      Bei .NET-Services gelten die folgenden Begrenzungen für Verzeichnisse:

      - Windows: 209 abzüglich der Länge des Servicenamens und der Erweiterung .asmx.
    - Das Implementierungsdienstprogramm prüft, ob die Begrenzung für APPLICID überschritten wird. Wenn die Begrenzung überschritten wird, versucht das Dienstprogramm nicht, den auslösenden Prozess zu definieren. Es zeigt eine Fehlernachricht an und der Implementierungsprozess schlägt fehl, ohne dass Implementierungsschritte ausgeführt werden.

    Die folgenden Beispiele zeigen die vom Implementierungsdienstprogramm generierten Konfigurations- und Startbefehle zum Starten eines SOAP-Empfangsprogramms von WebSphere MQ.

```
DEFINE PROCESS(requestQueue) APPLICID(applicIDStr) REPLACE
ALTER  QLOCAL (requestQueue) TRIGTYPE(FIRST) TRIGGER
       PROCESS(requestQueue) INITQ(initQueueName) TRIGMPRI(0)
```

Abbildung 15. WebSphere MQ-Konfigurationsbefehle zum Auslösen eines SOAP-Empfangsprogramms

```
applicIDStr = start "Java WMQSoapListener -requestQueue"
                    /min .\generated\server\startWMQJListener.cmd;
```

Abbildung 16. Starten eines Axis-SOAP-Empfangsprogramms unter Windows

```
applicIDStr = start "WMQAsmxListener -className\
                    /min .\generated\server\startWMQNListener.cmd;
```

Abbildung 17. Starten eines .NET-SOAP-Empfangsprogramms unter Windows

```
applicIDStr = xterm -iconic -T \"Java WMQSoapListener_requestQueue\"
                  -e ./generated/server/startWMQJListener.sh & #
```

Abbildung 18. Starten eines Axis-SOAP-Empfangsprogramms auf Systemen mit UNIX and Linux

amqwRegisterdotNet: IBM WebSphere MQ Transport for SOAP für .NET registrieren

IBM WebSphere MQ Transport for SOAP wird für den Global Assembly Cache von .NET registriert.

Verwendungszweck

amqwRegisterdotNet registriert den SOAP-Sender, das SOAP-Empfangsprogramm und den WSDL-Prozessor von WebSphere MQ für .NET Framework 1 oder 2.

Syntax

Beschreibung

amqwRegisterdotNet wird automatisch während der Installation ausgeführt. Der Befehl muss danach nicht noch einmal ausgeführt werden, wenn das von Ihnen verwendete .NET Framework vor WebSphere MQ Transport for SOAP installiert wurde. Sie können den Befehl beliebig oft ausführen, zum Beispiel, um WebSphere MQ Transport for SOAP für verschiedene Versionen von .NET Framework zu registrieren.

Anmerkung: Unter Windows 2003 Server müssen Sie außerdem das Dienstprogramm **aspnet_regiis** ausführen, auch wenn Sie nicht auf Internet Information Server (IIS) implementieren. Die Position des Dienstprogramms `aspnet_regiis.exe` kann je nach Version von Microsoft .NET Framework variieren, befindet sich jedoch normalerweise im Verzeichnis `%SystemRoot%/Microsoft.NET/Framework/version number/aspnet_regiis`. Wenn mehrere Versionen installiert sind, verwenden Sie **aspnet_regiis** für die Version von .NET Framework, die Sie einsetzen.

Apache-Softwarelizenz

Apache-Lizenz, Version 2.0, Januar 2004, http://www.apache.org/licenses/

http://www.apache.org/licenses/

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. bearbeitet wird.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and

subsequently incorporated within the Work.

2. Erteilung der Copyrightlizenz. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Erteilung einer Patentlizenz. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Umverteilung. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Einreichung von Beiträgen. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Marken. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Gewährleistungsausschluss. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Haftungsbegrenzung. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Gewährleistung oder zusätzliche Haftung akzeptieren. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

SOAP-Einstellungen für MQMD

Der IBM WebSphere MQ SOAP-Sender und der IBM WebSphere MQ SOAP-Listener erstellen einen Nachrichtendeskriptor (**MQMD**). In diesem Abschnitt werden die Felder beschrieben, die Sie im MQMD festlegen müssen, wenn Sie einen eigenen SOAP-Sender oder -Listener erstellen.

Verwendungszweck

Die im **MQMD** festgelegten Werte steuern den Nachrichtenaustausch zwischen dem SOAP-Sender von IBM WebSphere MQ, dem SOAP-Empfangsprogramm von IBM WebSphere MQ und dem SOAP-Clientprogramm. Wenn Sie einen eigenen SOAP-Sender oder ein eigenes SOAP-Empfangsprogramm erstellen, befolgen Sie die Regeln in Tabelle 584 auf Seite 993.

Beschreibung

In Tabelle 584 auf Seite 993 wird beschrieben, wie die **MQMD**-Felder vom SOAP-Sender von IBM WebSphere MQ und vom SOAP-Empfangsprogramm von IBM WebSphere MQ festgelegt werden. Wenn Sie einen eigenen Sender oder ein eigenes Empfangsprogramm schreiben, müssen Sie diese Felder entsprechend den Regeln für den Nachrichtenaustausch festlegen. Das SOAP-Empfangsprogramm von IBM WebSphere MQ entspricht den typischen Nachrichtenaustauschprotokollen von IBM WebSphere MQ. Wenn Sie einen eigenen Sender schreiben, der mit den SOAP-Empfangsprogrammen von IBM WebSphere MQ zusammenarbeiten soll, können Sie andere **MQMD**-Werte festlegen.

In Tabelle 584 auf Seite 993 sind die Werte in der Spalte Einstellung wie folgt aufgegliedert:

Anforderung, unidirektional
Einstellungen, die der SOAP-Sender von IBM WebSphere MQ festlegt.

Antwort, Bericht
Einstellungen, die das SOAP-Empfangsprogramm von IBM WebSphere MQ in Reaktion auf eine Anforderung des SOAP-Senders von IBM WebSphere MQ festlegt.

ALL
Einstellungen, die sowohl der SOAP-Sender von IBM WebSphere MQ als auch das SOAP-Empfangsprogramm von IBM WebSphere MQ festlegen.

Angepasster Sender
Sie können Ihren eigenen Sender schreiben. In der Regel überschreibt ein angepasster Sender die Standardberichtsoptionen.

Tabelle 584. SOAP-Einstellungen für MQMD

Name des Felds	Einstellung	Werte
StrucId	**ALL** MQMD_STRUC_ID	'MD¬¬'[1]
Version	**ALL** MQMD_VERSION_2	2
Report	**ALL** MQRO_NONE + MQRO_NEW_MSG_ID + MQRO_COPY_MSG_ID_TO_CORREL_ID + MQRO_EXCEPTION + MQRO_EXPIRY + MQRO_DISCARD **Angepasster Sender** Siehe „Angepasste Berichtsoptionen" auf Seite 997.	52428800
MsgType	**Anforderung** MQMT_REQUEST **Antwort** MQMT_REPLY **Bericht** MQMT_REPORT **Unidirektional** MQMT_DATAGRAM	**MQMT_REQUEST** 1 **MQMT_REPLY** 2 **MQMT_REPORT** 4 **MQMT_DATAGRAM** 8
Expiry	**Anforderung, unidirektional** In der Option für die Ablaufzeit (Expiry) im URI angegeben. Standardwert ist MQEI_UNLIMITED. **Antwort** Wert von Expiry in der Anforderungsnachricht. **Bericht** MQEI_UNLIMITED	**MQEI_UNLIMITED** -1

Tabelle 584. SOAP-Einstellungen für MQMD (Forts.)		
Name des Felds	**Einstellung**	**Werte**
Feedback	**Anforderung, Antwort, unidirektional** MQFB_NONE. **Bericht** • Von Warteschlangenmanager generiert - Wert wird gemäß den normalen Regeln festgelegt. • Vom SOAP-Empfangsprogramm von IBM WebSphere MQ generiert: **MQRC_BACKOUT_THRESHOLD_REACHED** Rücksetzschwellenwert für Mehrfachversuche überschritten. **MQRCCF_MD_FORMAT_ERROR** Es wird nicht erkannt, dass die Nachricht über einen MQRFH2-Header verfügt. **MQRC_RFH_PARM_MISSING** Ein erforderlicher Parameter, z. B. SoapAction, fehlt in **MQRFH2**. **MQRC_RFH_FORMAT_ERROR** Eine allgemeine Integritätsprüfung von **MQRFH2** wurde nicht bestanden, z. B. interne Längen beschädigt. **MQRC_RFH_ERROR** **MQRFH2** hat eine Integritätsprüfung bestanden, jedoch ist für den Nachrichtentext nicht MQFMT_NONE festgelegt.	**MQFB_NONE** 0 **MQRC_BACKOUT_THRESHOLD_REACHED** 2362 **MQRCCF_MD_FORMAT_ERROR** 3023 **MQRC_RFH_PARM_MISSING** 2339 **MQRC_RFH_FORMAT_ERROR** 2421 **MQRC_RFH_ERROR** 2334
Encoding	**ALL** MQENC_NATIVE	Von der Umgebung abhängig
CodedCharSetId	**ALL** UTF-8 festgelegt.	1208
Format	**Anforderung, Antwort, unidirektional** MQFMT_RF_HEADER_2 **Bericht** **Berichte des Warteschlangenmanagers.** Entspricht den IBM WebSphere MQ-Regeln. **Berichte des SOAP-Empfangsprogramms von IBM WebSphere MQ.** Format der ursprünglichen Anforderungsnachricht.	**MQFMT_RF_HEADER_2** "MQRFH2 "

Tabelle 584. SOAP-Einstellungen für MQMD (Forts.)

Name des Felds	Einstellung	Werte
Priority	**Anforderung, unidirektional** In der Option für die Priorität (Priority) im URI angegeben. Standardwert ist MQPRI_PRIORITY_AS_Q_DEF. **Antwort, Bericht** Wert von Priority in der Anforderungsnachricht.	**MQPRI_PRIORITY_AS_Q_DEF** -1
Persistence	**Anforderung, unidirektional** MQPER_PERSISTENCE_AS_Q_DEF. **Antwort, Bericht** Wert von Persistence in der Anforderungsnachricht.	**MQPER_PERSISTENCE_AS_Q_DEF** 2
MsgId	**Anforderung, unidirektional** Vom Warteschlangenmanager generiert. **Antwort, Bericht** Der SOAP-Sender von IBM WebSphere MQ legt MQRO_NEW_MSG_ID fest und *MsgId* wird generiert.	**Generiert** Warteschlangenmanager generiert einen eindeutigen Wert.
CorrelId	**Anforderung, unidirektional, Bericht** MQCI_NONE **Antwort, Bericht** Der SOAP-Sender von IBM WebSphere MQ legt MQRO_COPY_MSG_ID_TO_CORREL_ID fest und das Empfangsprogramm kopiert *MsgId* aus der Anforderungsnachricht.	**MQCI_NONE** 0
BackoutCount	**ALL** Nicht verwendet	0
ReplyToQ	**Anforderung** In der Option für das Antwortziel (replyDestination) im URI angegeben. Standardwert ist SYSTEM.SOAP.RESPONSE.QUEUE. **Antwort, unidirektional, Bericht** Bleibt leer.	
ReplyToQMgr	**ALL** Feld bleibt leer.	Vom Warteschlangenmanager generiert; weitere Informationen finden Sie unter Empfangswarteschlange für Antworten und Warteschlangenmanager.

Tabelle 584. SOAP-Einstellungen für MQMD (Forts.)		
Name des Felds	**Einstellung**	**Werte**
UserIdentifier	**Anforderung, unidirektional, Bericht** Bleibt leer. **Antwort** Richtet sich nach der Option -i *passContext*, die an das Empfangsprogramm übergeben wird, und nach der Berechtigung, mit der das Empfangsprogramm ausgeführt wird.	**Anforderung, unidirektional, Bericht** Vom Warteschlangenmanager generiert, siehe „UserIdentifier (MQCHAR12)" auf Seite 447. **Antwort** *Variable*
AccountingToken	**ALL** MQACT_NONE	**MQACT_NONE** Nullzeichenfolge oder Leerzeichen. Vom Warteschlangenmanager festgelegt, siehe „AccountingToken (MQBYTE32)" auf Seite 402.
ApplIdentityData	**ALL** --	Nullzeichenfolge oder Leerzeichen[2].
PutApplType	**ALL** MQAT_NO_CONTEXT	**MQAT_NO_CONTEXT** 0 Wert wird vom Warteschlangenmanager generiert, siehe „PutApplType (MQLONG)" auf Seite 433.
PutApplName	**ALL** --	Wert wird vom Warteschlangenmanager generiert, siehe „PutApplName (MQCHAR28)" auf Seite 431.
PutDate	**ALL** --	Wert wird vom Warteschlangenmanager generiert, siehe „PutDate (MQCHAR8)" auf Seite 434.
PutTime	**ALL** --	Wert wird vom Warteschlangenmanager generiert, siehe „PutTime (MQCHAR8)" auf Seite 435.
ApplOriginData	**ALL** --	Nullzeichenfolge oder Leerzeichen[2].
GroupId	**Anforderung, unidirektional, Bericht** MQGI_NONE **Antwort** Feld wird aus der Anforderungsnachricht kopiert.	Nullen

Tabelle 584. SOAP-Einstellungen für MQMD (Forts.)		
Name des Felds	**Einstellung**	**Werte**
MsgSeqNumber	**Anforderung, unidirektional, Bericht** Nicht verwendet **Antwort** Feld wird aus der Anforderungsnachricht kopiert.	Vom Warteschlangenmanager generiert, siehe Physische Reihenfolge in einer Warteschlange.
Offset	**Anforderung, unidirektional, Bericht** Nicht verwendet **Antwort** Feld wird aus der Anforderungsnachricht kopiert.	`0`
MsgFlags	**Anforderung, unidirektional, Bericht** `MQMF_NONE` **Antwort** Feld wird aus der Anforderungsnachricht kopiert.	**`MQMF_NONE`** `0` Siehe „MsgFlags (MQLONG)" auf Seite 420.
OriginalLength	**Anforderung, unidirektional, Antwort** `MQOL_UNDEFINED` **Bericht** Länge der ursprünglichen Anforderungsnachricht.	**`MQOL_UNDEFINED`** `-1`
Anmerkungen: 1. Das Symbol ¬ stellt ein einzelnes Leerzeichen dar. 2. Der Wert Nullzeichenfolge oder Leerzeichen bezeichnet die Nullzeichenfolge in C bzw. Leerzeichen in anderen Programmiersprachen.		

Angepasste Berichtsoptionen

Sie können Ihren eigenen SOAP-Sender schreiben und ihn mit den mitgelieferten Empfangsprogrammen verwenden. Mit einem eigenen Sender können Sie beispielsweise die Auswahl an Berichtsoptionen ändern. Die SOAP-Empfangsprogramme von IBM WebSphere MQ unterstützen die meisten Kombinationen von Berichtsoptionen, wie den folgenden Listen zu entnehmen ist.

- Von den SOAP-Empfangsprogrammen von IBM WebSphere MQ unterstützte Berichtsoptionen:
  - `MQRO_EXCEPTION`
  - `MQRO_EXCEPTION_WITH_DATA`
  - `MQRO_EXCEPTION_WITH_FULL_DATA`
  - `MQRO_DEAD_LETTER_Q`
  - `MQRO_DISCARD_MSG`
  - `MQRO_NONE`
  - `MQRO_NEW_MSG_ID`
  - `MQRO_PASS_MSG_ID`
  - `MQRO_COPY_MSG_ID_TO_CORREL_ID`
  - `MQRO_PASS_CORREL_ID`
- Vom Warteschlangenmanager unterstützte Berichtsoptionen:

  - MQRO_COA
  - MQRO_COA_WITH_DATA
  - MQRO_COA_WITH_FULL_DATA
  - MQRO_COD
  - MQRO_COD_WITH_DATA
  - MQRO_COD_WITH_FULL_DATA
  - MQRO_EXPIRATION
  - MQRO_EXPIRATION_WITH_DATA
  - MQRO_EXPIRATION_WITH_FULL_DATA
- Die folgenden Berichtsoptionen werden nicht von den SOAP-Empfangsprogrammen von IBM WebSphere MQ unterstützt.
  - MQRO_PAN
  - MQRO_NAN

Das Verhalten von SOAP-Empfangsprogrammen von IBM WebSphere MQ in Abhängigkeit von den Einstellungen für MQRO_EXCEPTION_* und MQRO_DISCARD ist in Tabelle 585 auf Seite 998 beschrieben.

Die Notation MQRO_EXCEPTION_* bedeutet, dass es sich um MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA oder MQRO_EXCEPTION_WITH_FULL_DATA handeln kann.

Tabelle 585. Verhalten des Empfangsprogramms in Abhängigkeit von den Einstellungen MQRO_EXCEPTION_* und MQRO_DISCARD

	MQRO_DISCARD aktiviert	**MQRO_DISCARD nicht aktiviert**
MQRO_EXCEPTION_* aktiviert	Standardverhalten. Bei Bedarf werden automatisch Berichtsnachrichten generiert. Die Originalanforderung wird verworfen. Wenn eine Berichtsnachricht nicht an die Antwortwarteschlange zurückgegeben werden konnte, wird sie an die Warteschlange für nicht zustellbare Nachrichten gesendet.	Bei Bedarf werden automatisch Berichtsnachrichten generiert. Die Originalanforderung wird an die Warteschlange für nicht zustellbare Nachrichten gesendet. Wenn die Berichtsnachricht nicht an die Antwortwarteschlange zurückgegeben werden konnte, wird sie ebenfalls an die Warteschlange für nicht zustellbare Nachrichten gesendet. In diesem Fall gibt es daher für die fehlgeschlagene Anforderung zwei Einträge in der Warteschlange für nicht zustellbare Nachrichten.
MQRO_EXCEPTION_* nicht aktiviert	Es werden keine Berichtsnachrichten automatisch generiert, wenn das eingehende Format nicht erkannt oder die Anzahl der Rücksetzversuche überschritten wird. Die Nachricht wird nicht an die Warteschlange für nicht zustellbare Nachrichten gesendet. An den Client wird keine Benachrichtigung zurückgegeben, die er prüfen könnte, und die ursprüngliche Anforderungsnachricht geht verloren.	Es werden keine Berichtsnachrichten automatisch generiert, wenn das eingehende Format nicht erkannt oder die Anzahl der Rücksetzversuche überschritten wird. Die ursprüngliche Anforderungsnachricht wird jedoch in die Warteschlange für nicht zustellbare Nachrichten geschrieben, wenn sonst ein Bericht generiert worden wäre.

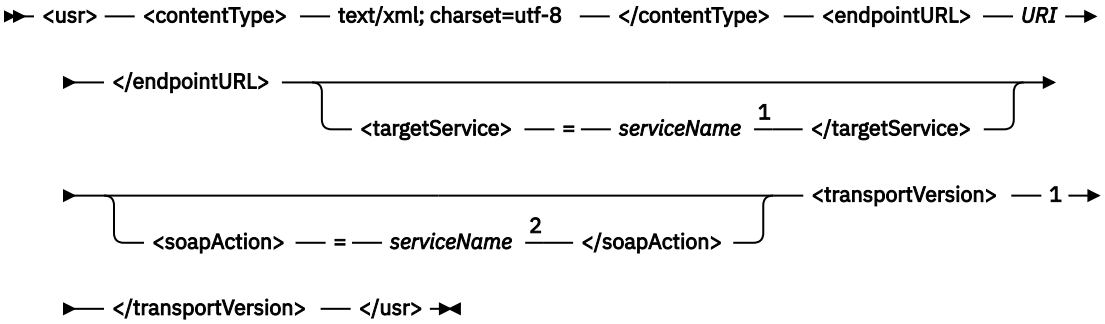
SOAP-Einstellungen für MQRFH2

Die SOAP-Sender und -Empfangsprogramme von IBM WebSphere MQ erstellen oder erwarten einen MQRFH2 mit den folgenden Einstellungen.

Verwendungszweck

Die WebSphere MQ -SOAP-Sender fügen Eigenschaften zum Ordner <usr> hinzu, der von WebSphere MQ JMS erstellt wurde. Die Eigenschaften enthalten Informationen, die der SOAP-Container in der Zielumgebung erfordert. „Eigenschaftensyntax" auf Seite 999 beschreibt die Syntax der Eigenschaften, wenn sie einem MQRFH2hinzugefügt werden. Die Beschreibung eines MQRFH2 -Headers finden Sie unter MQRFH2 -Rules and formatting header 2.

Eigenschaftensyntax



Anmerkungen:

[1] targetService ist erforderlich für .NET Framework 1 und 2 und wird bei Axis 1.4 nicht verwendet.
[2] soapAction ist für .NET Framework 1 und 2 optional und wird bei Axis 1.4 nicht verwendet.

Parameter

contentType
contentType enthält immer die Zeichenfolge `text/xml; charset=utf-8`.

endpointURL
Weitere Informationen finden Sie im Abschnitt „URI-Syntax und -Parameter für die Web-Service-Implementierung" auf Seite 1016.

targetService
[6]Auf Axis ist *serviceName* der vollständig qualifizierte Name eines Java -Service, z. B. `targetService=javaDemos.service.StockQuoteAxis`. Wenn targetService nicht angegeben ist, wird ein Service mit dem Axis-Standardmechanismus geladen.

[7]Unter .NET ist *serviceName* der Name eines .NET-Service, der sich im Implementierungsverzeichnis befindet, z. B. `targetService=myService.asmx`. In der .NET-Umgebung ermöglicht der Parameter targetService, dass ein einzelnes SOAP-Empfangsprogramm von WebSphere MQ Anforderungen für mehrere Services verarbeiten kann. Diese Services müssen im gleichen Verzeichnis implementiert sein.

soapAction

transportVersion
Für transportVersion ist immer 1 festgelegt.

[6] Nur Java-Service
[7] Nur .NET-Service

Beispiel

Das Beispiel zeigt einen MQRFH2 sowie die darauffolgende SOAP-Nachricht. Die Längen der Ordner werden dezimal angegeben.

Anmerkung: & ist im URI als & codiert.

```
52464820 00000002 000002B0 00000001 RFH 0002 1208 0001
000004B8 20202020 20202020 00000000 1208 0000
000004B8                           1208
32  <mcd>
        <Msd>jms_bytes</Msd>
    </mcd>?
208 <jms>
        <Dst>queue://queue://SOAPJ.demos</Dst>
        <Rto>queue://WMQSOAP.DEMO.QM/SYSTEM.SOAP.RESPONSE.QUEUE</Rto>
        <Tms>1157388516465</Tms>
        <Cid>ID:000000000000000000000000000000000000000000000000</Cid>
        <Dlv>1</Dlv>
    </jms>
400 <usr>
        <contentType>text/xml; charset=utf-8</contentType>
        <transportVersion>1</transportVersion>
        <endpointURL>
            jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
            &amp;connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)
                clientConnection(localhost%25289414%2529)
                clientChannel(TESTCHANNEL)
            &amp;replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
            &amp;initialContextFactory=com.ibm.mq.jms.Nojndi
        </endpointURL>
    </usr>
<?xml version="1.0" encoding="UTF-8"?>
    <soapenv:Envelope
            xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsd="https://www.w3.org/2001/XMLSchema"
            xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
        <soapenv:Body>
            <ns1:getQuote
                    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    xmlns:ns1="soap.server.StockQuoteAxis_Wmq">
                <in0 xsi:type="xsd:string">XXX</in0>
            </ns1:getQuote>
        </soapenv:Body>
    </soapenv:Envelope>
```

runivt: Installationsprüftest für WebSphere MQ Transport for SOAP

Zum Lieferumfang von IBM WebSphere MQ Transport for SOAP gehört ein Installationsprüftest. **runivt** führt eine Reihe von Demo-Anwendungen aus und stellt sicher, dass die Umgebung nach der Installation ordnungsgemäß konfiguriert ist.

Verwendungszweck

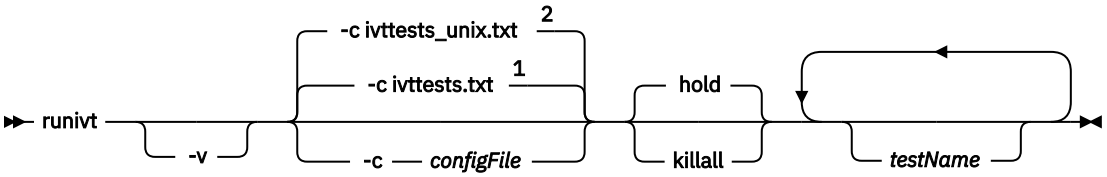
Der Befehl **runivt** verwendet die in WebSphere MQ Transport for SOAP enthaltenen Musterprogramme, um Web-Service-Anforderungen von Clients an Services zu senden. Es führt Tests für Axis 1.4, .NET Framework 1 und .NET Framework 2 aus. Die Tests werden in einer Testscriptdatei konfiguriert. Die Standardtestscriptdatei für Windows führt eine Kombination von Tests zwischen Java-und .NET-Clients und -Services aus.

Beschreibung

runivt muss aus seinem eigenen Verzeichnis heraus ausgeführt werden.

Der Befehl startet Empfangsprogramme in einem separaten Befehlsfenster. Aus diesem Grund müssen Sie den Befehl auf Systemen mit UNIX and Linux in einer X Window System-Sitzung ausführen.

`runivt` syntax



Anmerkungen:

[1] Default on Windows
[2] Default on UNIX and Linux systems

Parameter für `runivt`

`-v`
Ausführlicher Modus. Dabei werden detailliertere Fehlernachrichten an die Konsole gesendet.

`-c` ***configFile***
Eine Konfigurationsdatei, die die auszuführenden Tests definiert. Standardmäßig wird die Standardkonfigurationsdatei verwendet, die mit Windows-, UNIX -oder Linux -Systemen bereitgestellt wird.

`hold`
Nach Abschluss des Tests Empfangsprogramm weiter ausführen.

`killall`
Nach Abschluss des Tests Empfangsprogramm beenden.

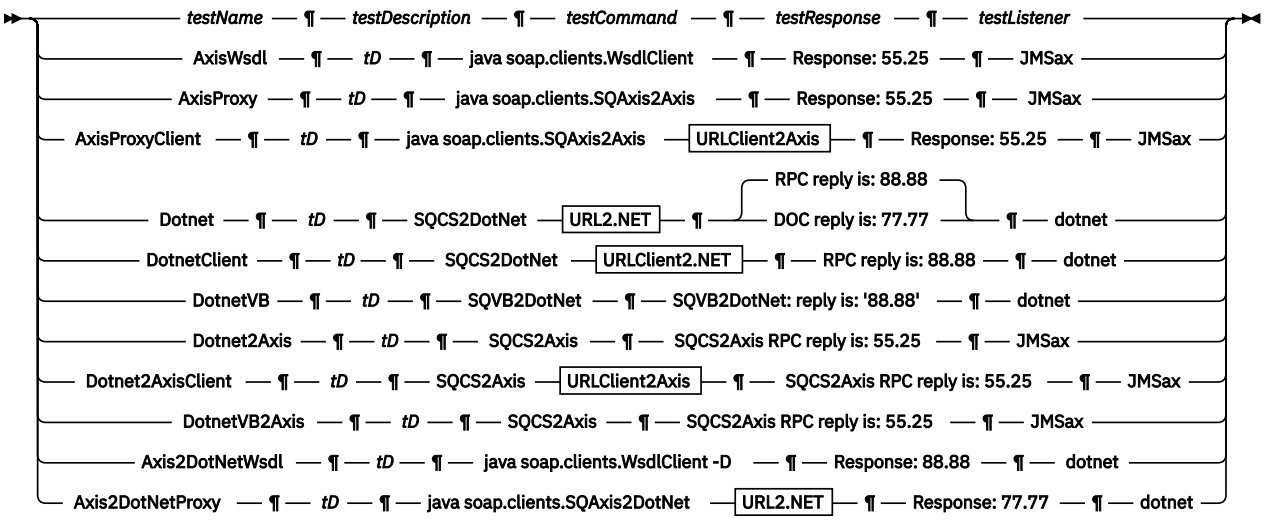
testName
Eine durch Leerzeichen getrennte Liste der auszuführenden Tests. Die Testnamen werden aus der Konfigurationsdatei ausgewählt. Wenn keine Namen angegeben sind, werden alle Tests in der Konfigurationsdatei ausgeführt.

Configuration file

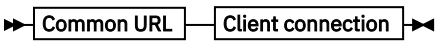
Each configuration file parameter is a separate line of the file. Leave a blank line between each group of parameters.

The parameters in the `ivttests.txt` parameter file are listed.

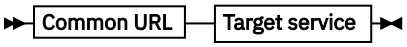
***configFile* syntax**



URLClient2Axis

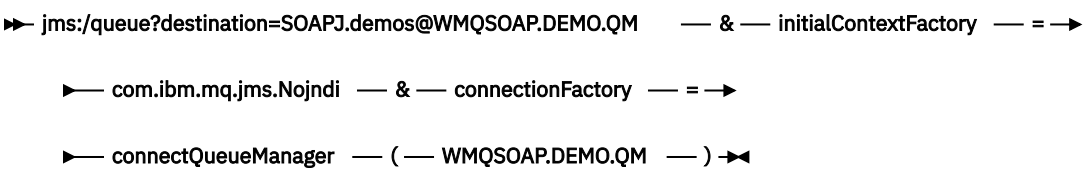


URL2.NET

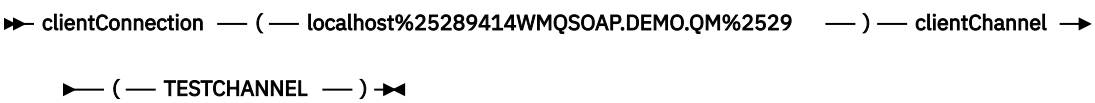


URLClient2.NET

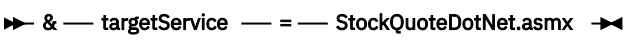
Common URL



Client connection



Target service



configFile-Parameter

testName
Der Name des Tests. Verwenden Sie *testName* im Befehl **runivt**.

testDescription
Dokumentation zum Test

testCommand
Der Befehl, den der Befehl **runivt** ausführt, um die Clientanforderung zu tätigen.

testResponse
Die exakte Antwortzeichenfolge, die von der Clientanforderung an die Konsole zurückgegeben wird. Damit der Test erfolgreich ist, muss die *testResponse* mit der tatsächlichen Antwort übereinstimmen.

testListener
Der Name des SOAP-Empfangsprogramms von WebSphere MQ, das von **runivt** gestartet wird, um die SOAP-Anforderung zu verarbeiten. dotnet und JMSax sind Synonyme für die bereitgestellten Empfangsprogramme, **amqwSOAPNETlistener** und **SimpleJavaListener**.

Beispiele

```
runivt
```

Abbildung 19. Alle Standardtests ausführen

```
runivt dotnet
```

Abbildung 20. Einen bestimmten Test aus der Liste der Standardtests ausführen

```
runivt -c mytests.txt
```

Abbildung 21. Eine Reihe von benutzerdefinierten Tests ausführen

Zugehörige Informationen

WebSphere MQ Transport for SOAP überprüfen

Sichere Web-Services über IBM WebSphere MQ Transport for SOAP

Web-Services, die IBM WebSphere MQ Transport for SOAP verwenden, können auf zweierlei verschiedene Arten gesichert werden. Wahlweise können Sie einen SSL-Kanal zwischen Client und Server einrichten oder aber Web Services Security verwenden.

SSL und WebSphere MQ Transport for SOAP

WebSphere MQ Transport for SOAP stellt verschiedene SSL-Optionen bereit, die für die Verwendung mit Clientkanälen angegeben werden können, die für die Ausführung im SSL-Modus konfiguriert sind. Die Optionen unterscheiden sich je nach .NET-und Java-Umgebung. Der Sender und das Empfangsprogramm von WebSphere MQ SOAP verarbeiten nur die SSL-Optionen, die auf die jeweilige Umgebung anwendbar sind. Nicht anwendbare Optionen werden ignoriert.

Das Vorhandensein oder Fehlen der Option `sslCipherSpec` für .NET-Clients und der Option `sslCipherSuite` für Java-Clients bestimmt, ob SSL verwendet wird oder nicht. Wenn die Option im URI nicht angegeben wird, wird SSL standardmäßig nicht verwendet und alle anderen SSL-Optionen werden ignoriert. Wenn nicht anders angegeben, sind alle SSL-Optionen optional.

Legen Sie die SSL-Attribute für WebSphere MQ-Clients in der URI- oder Kanaldefinitionstabelle fest. Legen Sie auf dem Server die Attribute mithilfe der Funktionen von WebSphere MQ fest.

Die WebSphere MQ SSL-Standardoption `SSLCAUTH` wird standardmäßig festgelegt, wenn SSL auf dem Kanal aktiviert wird. Clients müssen sich selbst authentifizieren, damit die SSL-Kommunikation beginnen kann. Wenn `SSLCAUTH` nicht festgelegt wird, wird die SSL-Kommunikation ohne Clientauthentifizierung eingerichtet.

Damit sie sich selbst authentifizieren können, muss Clients ein Zertifikat im Schlüsselrepository zugewiesen sein, das vom Warteschlangenmanager akzeptiert wird. Für ein höheres Maß an Sicherheit können Sie WebSphere MQ-Kanäle so konfigurieren, dass nur Zertifikate von einer eingeschränkten Liste akzeptiert werden. Die Liste ist insofern eingeschränkt, als der definierte Name des Zertifikats anhand des Peer-Attributnamens des Kanals überprüft wird.

Wenn Sie Java verwenden, bewirkt die erste SSL-Verbindung von einem WebSphere MQ -SOAP-Client, dass die folgenden SSL-Parameter korrigiert werden. Dieselben Werte werden in nachfolgenden Verbindungen unter Verwendung desselben Clientprozesses verwendet:

- `sslKeyStore`
- `sslKeyStorePassword`
- `sslTrustStore`
- `sslTrustStorePassword`
- `sslFipsRequired`
- `sslLDAPCRLservers`

Welche Auswirkungen es hat, wenn diese Parameter in nachfolgenden Verbindungen von diesem Client aus geändert werden, ist nicht definiert.

Wenn Sie .NET verwenden, werden bei der ersten SSL-Verbindung von einem WebSphere MQ SOAP-Client aus die folgenden SSL-Parameter festgelegt. Dieselben Werte werden in nachfolgenden Verbindungen unter Verwendung desselben Clientprozesses verwendet:

- `SSLKeyRepository`
- `SSLCryptoHardware`
- `sslFipsRequired`
- `sslLDAPCRLservers`

Welche Auswirkungen es hat, wenn diese Parameter in nachfolgenden Verbindungen von diesem Client aus geändert werden, ist nicht definiert. Diese Parameter werden zurückgesetzt, wenn alle SSL-Verbindungen inaktiv werden und eine neue SSL-Verbindung hergestellt wird.

Die folgenden Eigenschaften können auch als Systemeigenschaften angegeben werden:

- `sslKeyStore`
- `sslKeyStorePassword`
- `sslTrustStore`
- `sslTrustStorePassword`

Wenn sie sowohl als Systemeigenschaften als auch im URI angegeben werden und die Werte unterschiedlich sind, zeigt das Implementierungsdienstprogramm eine Warnung an. Die URI-Werte haben Vorrang.

Zugehörige Tasks

Angeben, dass nur FIPS-zertifizierte CipherSpecs während der Ausführung auf dem MQI-Client verwendet werden

Zugehörige Verweise

Parameter von SSL-Verbindungsfactorys im URI von WebSphere MQ Web Services
Fügen Sie SSL-Optionen zur Liste mit Optionen von Verbindungsfactorys im URI von IBM WebSphere MQ Web Services hinzu.

Federal Information Processing Standards (FIPS) für UNIX, Linux und Windows

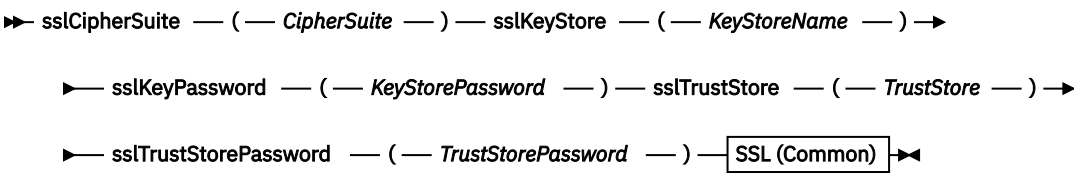
Parameter von SSL-Verbindungsfactorys im URI von WebSphere MQ Web Services

Fügen Sie SSL-Optionen zur Liste mit Optionen von Verbindungsfactorys im URI von IBM WebSphere MQ Web Services hinzu.

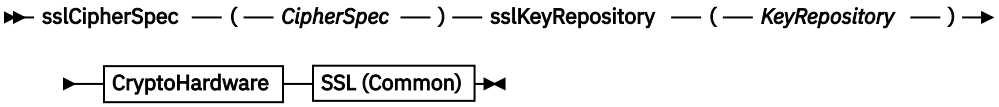
Verwendungszweck

Sie können eine sichere Verbindung zwischen einem IBM WebSphere MQ Web Services-Client und dem Warteschlangenmanager verwenden, der den Web-Service hostet. Die SSL-Optionen bestimmen, wie SSL für die Kanalverbindung zwischen dem IBM WebSphere MQ MQI-Client und -Server konfiguriert wird.

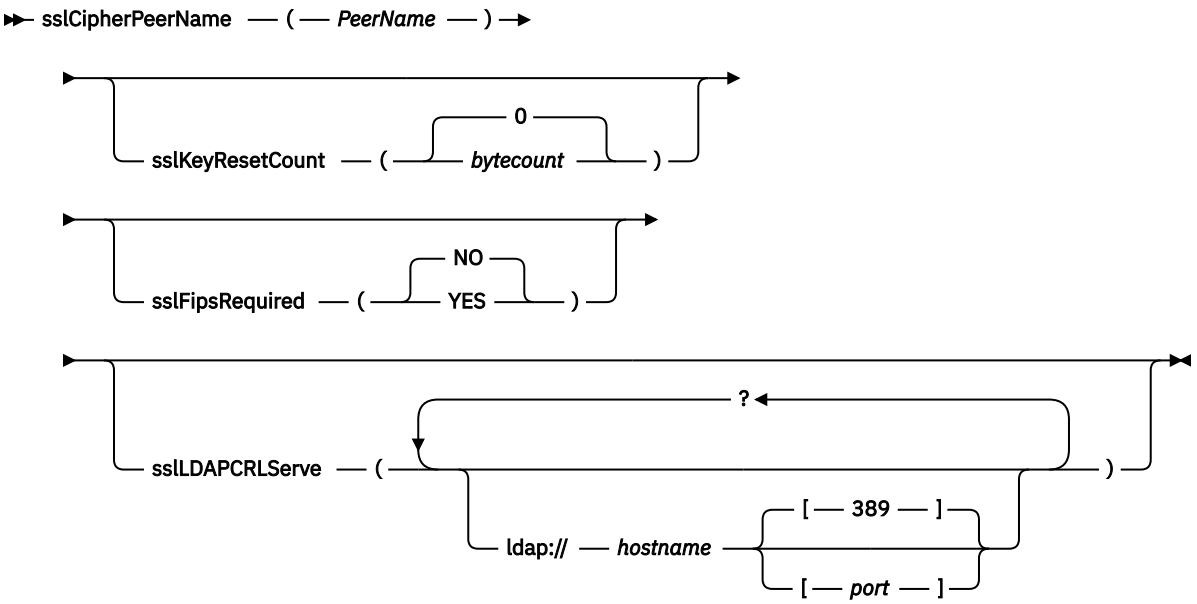
Syntax diagram
SSL (Java)



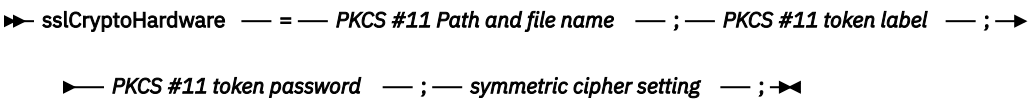
SSL (.NET)



SSL (Common)



CryptoHardware



Erforderliche SSL-Parameter (allgemein)

sslPeerName(*peerName*)
peerName gibt den auf dem Kanal verwendeten sslPeerName an.

Erforderliche SSL-Parameter (Java)

sslCipherSuite(*CipherSuite*)
CipherSuite gibt die auf dem Kanal verwendete sslCipherSuite an. Die vom Client angegebene Cipher-Suite muss mit der auf dem Serververbindungskanal angegebenen Cipher-Suite übereinstimmen.

sslKeyStore(*KeyStoreName*)
KeyStoreName gibt den auf dem Kanal verwendeten sslKeyStoreName an. Im Keystore ist der private Schlüssel des Clients gespeichert, der verwendet wird, um den Client für den Server zu authentifizieren. Der Keystore ist optional, wenn die SSL-Verbindung so konfiguriert wurde, dass anonyme Clientverbindungen akzeptiert werden.

sslKeyStorePassword(*KeyStorePassword*)
KeyStorePassword gibt das auf dem Kanal verwendete sslKeyStorePassword an.

sslTrustStore(*TrustStoreName*)
TrustStoreName gibt den auf dem Kanal verwendeten sslTrustStoreName an. Im Truststore ist das öffentliche Zertifikat des Servers oder die zugehörige Schlüsselkette gespeichert, die verwendet wird, um den Server für den Client zu authentifizieren. Der Truststore ist optional, wenn das Stammzertifikat einer Zertifizierungsstelle zum Authentifizieren des Servers verwendet wird. In Java werden Stammzertifikate im JRE-Zertifikatsspeicher `cacerts` gespeichert.

sslTrustStorePassword(*TrustStorePassword*)
TrustStorePassword gibt das auf dem Kanal gespeicherte sslTrustStorePassword an.

Erforderliche SSL-Parameter (.NET)

sslCipherSpec(*CipherSpec*)
CipherSpec gibt die auf dem Kanal verwendete sslCipherSpec an. Wenn die Option angegeben wurde, wird SSL auf dem Clientkanal verwendet.

sslKeyRepository(*KeyRepository*)
KeyRepository gibt die auf dem Kanal verwendete sslCipherSpec an, in der SSL-Schlüssel und -Zertifikate gespeichert werden. *KeyRepository* wird im Stammformat angegeben, d. h. mit vollständigem Pfad und Dateiname, jedoch ohne Dateierweiterung. Die Festlegung von sslKeyRepository hat dieselben Auswirkungen wie die Festlegung des Felds KeyRepository in der **MQSCO**-Struktur in einem MQCONNX-Aufruf.

Optionale SSL-Parameter (.NET)

sslCryptoHardware(*CryptoHardware*)
CryptoHardware gibt die auf dem Kanal verwendete sslCryptoHardware an. Die möglichen Werte für dieses Feld und die Auswirkungen der Festlegung dieses Felds sind dieselben wie für das Feld CryptoHardware der **MQSCO**-Struktur in einem MQCONNX-Aufruf.

Optionale SSL-Parameter (allgemein)

sslKeyResetCount(*bytecount*)
bytecount gibt die Anzahl der Byte an, die über einen SSL-Kanal übergeben werden, bevor der geheime SSL-Schlüssel neu vereinbart werden muss. Um die Neuvereinbarung von SSL-Schlüsseln zu inaktivieren, geben Sie für das Feld nichts oder null an. Null ist der einzige Wert, der in einigen Umgebungen unterstützt wird. Weitere Informationen finden Sie unter Geheime Schlüssel in WebSphere MQ Classes for Java neu vereinbaren. Die Festlegung von sslKeyResetCount hat dieselben Auswirkungen wie die Festlegung des Felds KeyResetCount in der **MQSCO**-Struktur in einem MQCONNX-Aufruf.

sslFipsRequired(*fipsCertified*)
fipsCertified gibt an, ob *CipherSpec* oder *CipherSuite* die FIPS-zertifizierte Verschlüsselung in IBM WebSphere MQ auf dem Kanal verwenden muss. Die Festlegung von *fipsCertified* hat dieselben Auswirkungen wie die Festlegung des Felds FipsRequired der **MQSCO**-Struktur in einem MQCONNX-Aufruf.

sslLDAPCRLServers(*LDAPServerList*)

LDAPServerList gibt eine Liste mit LDAP-Servern an, die für die Überprüfung der Zertifikatswiderrufslisten verwendet werden sollen.

Bei SSL-fähigen Clientverbindungen ist *LDAPServerList* eine Liste mit LDAP-Servern, die für die Überprüfung der Zertifikatswiderrufslisten verwendet werden sollen. Das vom Warteschlangenmanager bereitgestellte Zertifikat wird mit einem der aufgelisteten LDAP-CRL-Servern abgeglichen; wenn es gefunden wird, schlägt die Verbindung fehl. Alle LDAP-Server werden der Reihe nach ausprobiert, bis eine funktionsfähige Verbindung mit einem Server hergestellt werden kann. Wenn mit keinem Server eine Verbindung hergestellt werden kann, wird das Zertifikat zurückgewiesen. Nach dem erfolgreichen Herstellen einer Verbindung mit einem Server wird das Zertifikat abhängig von den diesem LDAP-Server vorgelegten Zertifikatswiderrufslisten (CRLs, Certificate Revocation Lists) akzeptiert oder zurückgewiesen.

Wenn *LDAPServerList* leer ist, wird das Zertifikat für den Warteschlangenmanager nicht mit einer Zertifikatswiderrufsliste abgeglichen. Es wird eine Fehlernachricht angezeigt, wenn die bereitgestellte Liste mit LDAP-URIs ungültig ist. Die Festlegung dieses Felds hat dieselben Auswirkungen wie die Einbeziehung von MQAIR-Datensätzen und der Zugriff auf diese über eine **MQSCO**-Struktur in einem MQCONNX-Aufruf.

Zugehörige Tasks
Angeben, dass nur FIPS-zertifizierte CipherSpecs während der Ausführung auf dem MQI-Client verwendet werden

Zugehörige Verweise

SSL und WebSphere MQ Transport for SOAP

WebSphere MQ Transport for SOAP stellt verschiedene SSL-Optionen bereit, die für die Verwendung mit Clientkanälen angegeben werden können, die für die Ausführung im SSL-Modus konfiguriert sind. Die Optionen unterscheiden sich je nach .NET-und Java-Umgebung. Der Sender und das Empfangsprogramm von WebSphere MQ SOAP verarbeiten nur die SSL-Optionen, die auf die jeweilige Umgebung anwendbar sind. Nicht anwendbare Optionen werden ignoriert.

Federal Information Processing Standards (FIPS) für UNIX, Linux und Windows

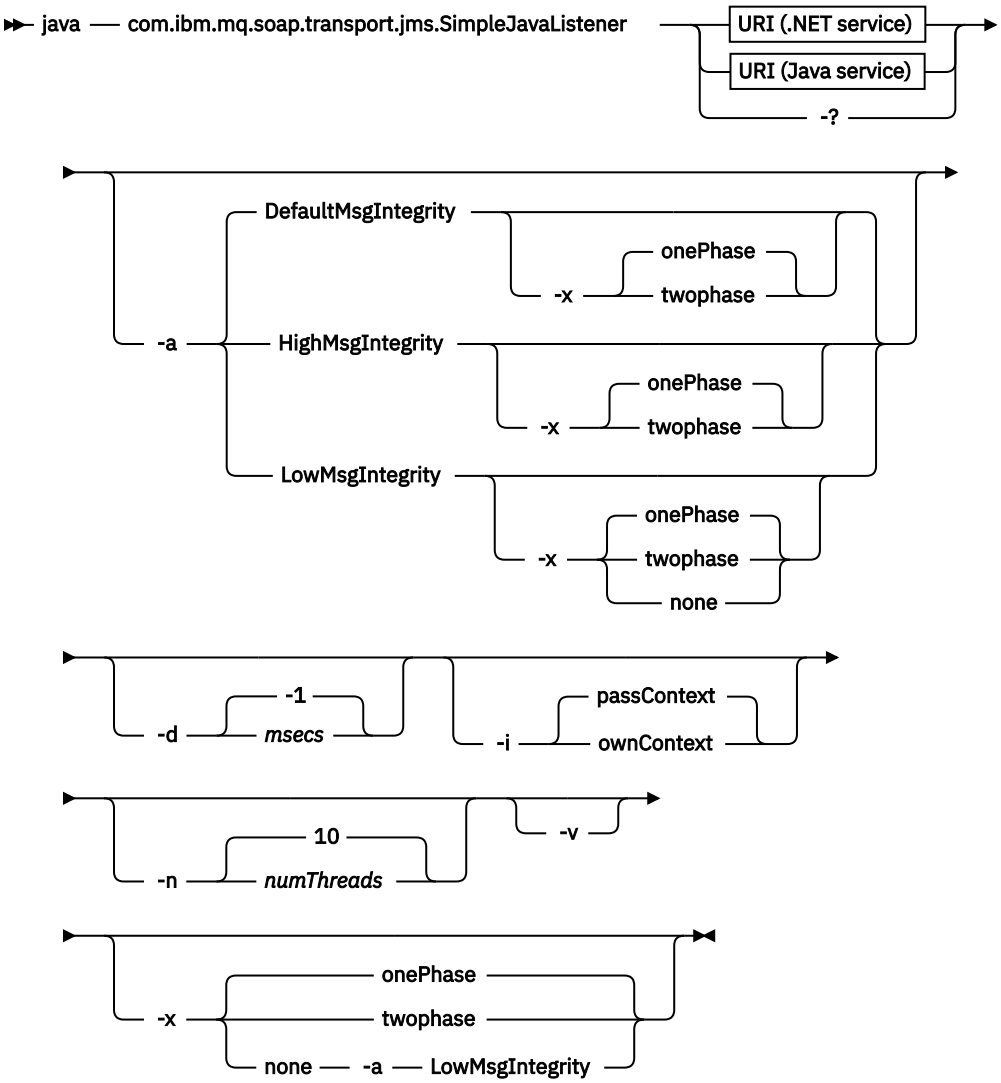
`SimpleJavaListener`: SOAP-Empfangsprogramm von IBM WebSphere MQ für Axis 1.4

Syntax und Parameter des SOAP-Empfangsprogramms von IBM WebSphere MQ für Axis 1.4.

Verwendungszweck

Startet das SOAP-Empfangsprogramm von IBM WebSphere MQ für Axis 1.4.

Java



Erforderliche Parameter

URI ***platform***
Siehe „URI-Syntax und -Parameter für die Web-Service-Implementierung” auf Seite 1016.

-?
Hilfetext ausgeben, der beschreibt, wie der Befehl verwendet wird.

Optionale Parameter

-a ***integrityOption***
integrityOption gibt das Verhalten von SOAP-Empfangsprogrammen von WebSphere MQ an, wenn es nicht möglich ist, eine fehlgeschlagene Anforderungsnachricht in die Warteschlange für nicht zustellbare Nachrichten einzureihen. *integrityOption* kann einen der folgenden Werte haben:

DefaultMsgIntegrity
Bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung mit der ursprünglichen, verworfenen Nachricht fort. Bei persistenten Nachrichten zeigt es eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet. DefaultMsgIntegrity gilt, wenn die Option -a ausgelassen wird oder wenn *integrityOption* nicht angegeben ist.

LowMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung fort, wobei die Nachricht verworfen wird.

HighMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Wenn -x none angegeben ist, muss -a LowMsgIntegrity angegeben sein. Wenn die Flags nicht kompatibel sind, wird das Implementierungsdienstprogramm mit einer Fehlernachricht beendet, ohne dass irgendein Implementierungsschritt ausgeführt wurde.

-d ***msecs***
msecs gibt an, wie lange (in Millisekunden) das SOAP-Empfangsprogramm von WebSphere MQ aktiv bleibt, wenn in einem beliebigen Thread Nachrichten eingegangen sind. Wenn für *msecs* der Wert -1 festgelegt ist, bleibt das Empfangsprogramm für unbegrenzte Zeit aktiv.

-i ***Context***
Context gibt an, ob die Empfangsprogramme den Identitätskontext übergeben. *Context* kann die folgenden Werte haben:

passContext
Identitätskontext der ursprünglichen Anforderungsnachricht in die Antwortnachricht einfügen. Das SOAP-Empfangsprogramm prüft, ob es berechtigt ist, den Kontext aus der Anforderungswarteschlange zu speichern und an die Antwortwarteschlange zu übergeben. Es führt diese Prüfungen zur Laufzeit aus, wenn es die Anforderungswarteschlange zum Speichern des Kontextes und die Antwortwarteschlange zum Übergeben des Kontextes öffnet. Wenn es nicht über die erforderliche Berechtigung verfügt oder der MQOPEN-Aufruf fehlschlägt, wird die Antwortnachricht nicht verarbeitet. Die Antwortnachricht wird in die Warteschlange für nicht zustellbare Nachrichten eingereiht, wobei der Header der nicht zustellbaren Nachricht den Rückkehrcode aus dem fehlgeschlagenen MQOPEN-Aufruf enthält. Das Empfangsprogramm fährt anschließend mit der normalen Verarbeitung der nachfolgenden eingehenden Nachrichten fort.

ownContext
Das SOAP-Empfangsprogramm übergibt keinen Kontext. Der zurückgegebene Kontext gibt die Benutzer-ID an, unter der das Empfangsprogramm ausgeführt wird, und nicht die Benutzer-ID, die die ursprüngliche Anforderungsnachricht erstellt hat.

Die Felder im Ursprungskontext werden vom Warteschlangenmanager festgelegt, nicht vom SOAP-Empfangsprogramm.

-n ***numThreads***
numThreads gibt die Anzahl der Threads in den generierten Startscripts für das SOAP-Empfangsprogramm von WebSphere MQ an. Der Standardwert ist 10. Es kann sinnvoll sein, diesen Wert zu erhöhen, wenn der Nachrichtendurchsatz hoch ist.

-v
-v legt die ausführliche Ausgabe von externen Befehlen fest. Fehlernachrichten werden immer angezeigt. Verwenden Sie -v, um Befehle auszugeben, die Sie anpassen können, um angepasste Implementierungsscripts zu erstellen.

-w ***serviceDirectory***
serviceDirectory ist das Verzeichnis, das den Web-Service enthält.

-x ***transactionality***
transactionality gibt den Typ der Transaktionssteuerung für das Empfangsprogramm an. Für *transactionality* kann einer der folgenden Werte festgelegt sein:

onePhase
Es wird die Einphasenunterstützung von IBM WebSphere MQ verwendet. Wenn das System während der Verarbeitung ausfällt, wird die Anforderungsnachricht erneut der Anwendung zugestellt. WebSphere MQ -Transaktionen stellen sicher, dass die Antwortnachrichten genau einmal geschrieben werden.

twoPhase
Die Zweiphasenunterstützung wird verwendet. Wenn der Service entsprechend geschrieben wurde, wird die Nachricht genau einmal zugestellt, in Koordination mit anderen Ressourcen und innerhalb einer einzelnen festgeschriebenen Ausführung des Service. Diese Option gilt nur für Verbindungen mit Serverbindung.

none
Keine Transaktionsunterstützung. Wenn das System während der Verarbeitung ausfällt, kann die Anforderungsnachricht verloren gehen, auch wenn sie persistent ist. In diesem Fall ist es möglich, dass der Service ausgeführt wurde oder nicht und dass Antwort-, Berichts- oder nicht zustellbare Nachrichten geschrieben wurden oder nicht.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Weitere Informationen finden Sie in der Beschreibung zum Flag -a.

Java-Beispiel

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

SOAP-Empfangsprogramme von WebSphere MQ

Ein SOAP-Empfangsprogramm von WebSphere MQ liest eine eingehende SOAP-Anforderung aus der Warteschlange, die im URI als Ziel angegeben ist. Es prüft das Format der Anforderungsnachricht und ruft dann über die Web-Services-Infrastruktur einen Web-Service auf. Ein SOAP-Empfangsprogramm von WebSphere MQ gibt alle Antworten oder Fehler von Web-Services über die im URI als Ziel angegebene Antwortwarteschlange zurück. Es gibt WebSphere MQ-Berichte an die Antwortwarteschlange zurück.

Der Begriff "Empfangsprogramm" wird hier in seiner Standardbedeutung in Bezug auf Web-Services verwendet. Es handelt sich nicht um das WebSphere MQ-Standardempfangsprogramm, das vom Befehl **runmqlsr** ausgegeben wird.

Beschreibung

Der Java-SOAP-Listener wird als Java-Klasse implementiert und führt Services mit Axis 1.4aus. Das .NET-Empfangsprogramm ist eine Konsolenanwendung und führt .NET Framework 1- oder .NET Framework 2-Services aus. Für .NET Framework 3-Services verwenden Sie den angepassten WebSphere MQ-Kanal für Microsoft Windows Communication Foundation (WCF).

Das Implementierungsdienstprogramm erstellt Scripts zum automatischen Starten von SOAP-Empfangsprogrammen für Java oder .NET. Ein SOAP-Empfangsprogramm kann manuell entweder über den Befehl **amqSOAPNETListener** oder durch Aufrufen der Klasse `SimpleJavaListener` gestartet werden. Sie können das SOAP-Empfangsprogramm von WebSphere MQ so konfigurieren, dass es als WebSphere MQ-Service gestartet wird, indem Sie die Option -s im Implementierungsdienstprogramm festlegen. Alternativ dazu können Sie Empfangsprogramme mithilfe einer Auslösefunktion starten oder die Empfangsprogrammscripts zum Starten und Beenden verwenden, die vom Implementierungsdienstprogramm generiert werden. Sie können die Auslösefunktion manuell konfigurieren oder die Implementierungsoptionen -tmq und -tmp verwenden, damit sie automatisch konfiguriert wird. Sie können ein Empfangsprogramm beenden, indem Sie für die Anforderungswarteschlange `GET(DISABLED)` festlegen.

Tabelle 586. Vom Implementierungsdienstprogramm generierte Befehlsscripts

Web-Service-Infrastruktur	UNIX and Linux-Systeme	Windows -Java	Windows .NET
Listener starten	**startWMQJListener.sh**	**startWMQJListener.cmd**	**startWMQNListener.cmd**
Empfangsprogramm stoppen	**endWMQJListener.sh**	**endWMQJListener.cmd**	**endWMQNListener.cmd**
Empfangsprogrammservice definieren	**defineWMQJListener.sh**	**defineWMQJListener.cmd**	**defineWMQNListener.cmd**

Das SOAP-Empfangsprogramm von WebSphere MQ übergibt die Felder `endpointURL` und `soapAction` aus der SOAP-Nachricht an die SOAP-Infrastruktur. Das Empfangsprogramm ruft den Service über die Web-Services-Infrastruktur auf und wartet auf die Antwort. Das Empfangsprogramm prüft `endpointURL` und `soapAction` nicht. Die Felder werden durch den SOAP-Sender von WebSphere MQ entsprechend den Daten festgelegt, die in dem von einem SOAP-Client festgelegten URI enthalten sind.

Das Empfangsprogramm erstellt die Antwortnachricht und sendet sie an das Antwortziel, das im URI der Anforderungsnachricht angegeben ist. Zusätzlich legt das Empfangsprogramm die Korrelations-ID in der Antwortnachricht entsprechend der Berichtsoption in der Anforderungsnachricht fest. Es gibt die Einstellungen für die Ablaufzeit, die Persistenz und die Priorität aus der Anforderungsnachricht zurück. Das Empfangsprogramm sendet außerdem unter bestimmten Umständen Berichtsnachrichten an Clients zurück.

Liegen in der SOAP-Anforderung Formatfehler vor, gibt das Empfangsprogramm über die als Ziel angegebene Antwortwarteschlange eine Berichtsnachricht an den Client zurück. Der Warteschlangenmanager gibt ebenfalls über die als Ziel angegebene Antwortwarteschlange Berichtsnachrichten an den Client zurück, wenn ein Bericht angefordert wurde. In Reaktion auf einige Ereignisse werden vollständige Berichtsnachrichten in die Antwortwarteschlange geschrieben:

- Ausnahmebedingung
- Nachricht abgelaufen
- Format der Anforderungsnachricht wird nicht erkannt
- Header **MQRFH2** hat die Integritätsprüfung nicht bestanden
- Format des Nachrichtenhauptteils ist nicht `MQFMT_NONE`
- Rücksetz-/Wiederholungsschwellenwert wurde bei der Verarbeitung der Anforderung durch das SOAP-Empfangsprogramm von WebSphere MQ überschritten

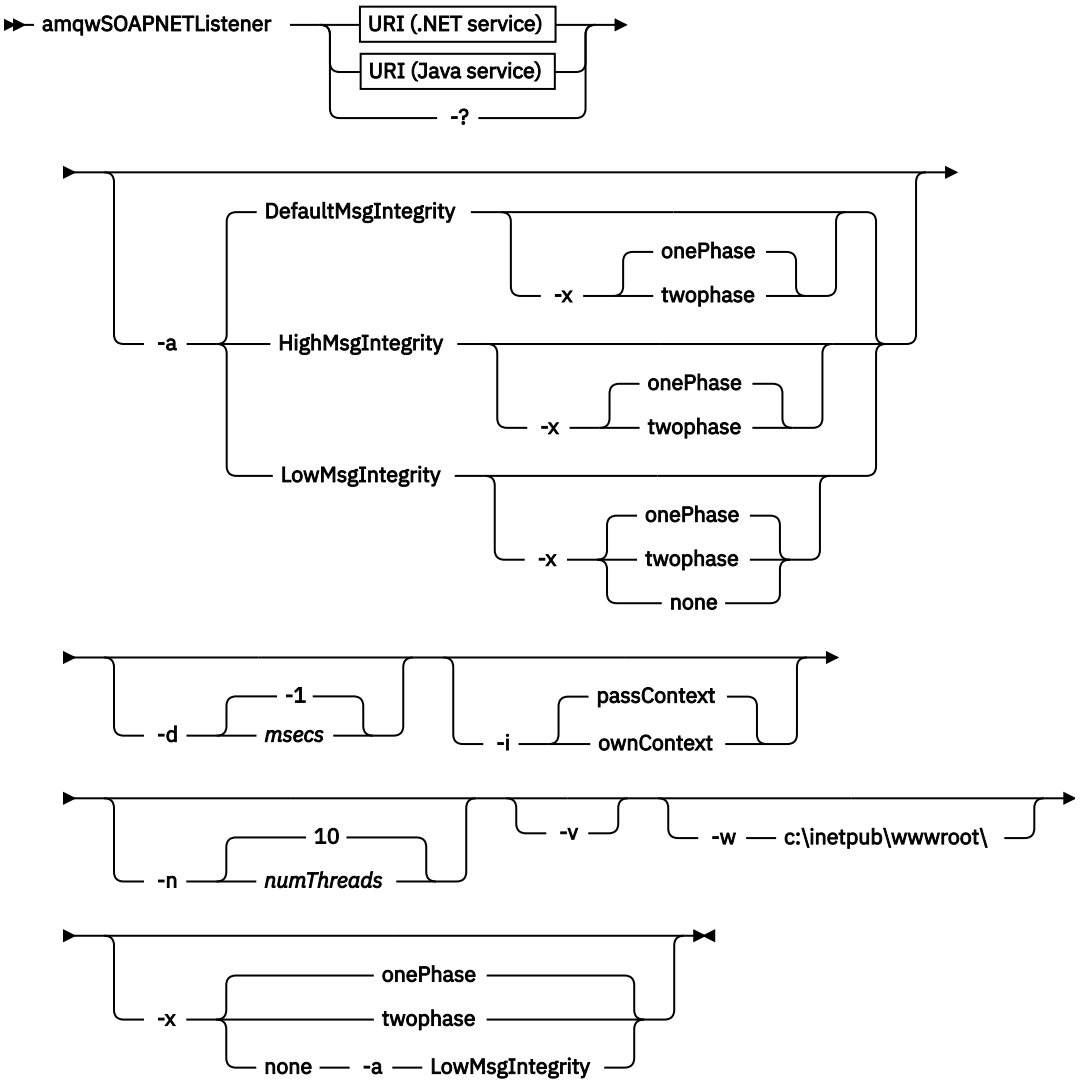
Der SOAP-Sender von WebSphere MQ legt die Berichtsoptionen `MQRO_EXCEPTION_WITH_FULL_DATA` und `MQRO_EXPIRATION_WITH_FULL_DATA` fest. Aufgrund der vom SOAP-Sender von WebSphere MQ festgelegten Berichtsoptionen enthält die Berichtsnachricht die gesamte ursprüngliche Anforderungsnachricht. Der Sender von WebSphere MQ SOAP legt auch die Option `MQRO_DISCARD` fest, die dazu führt, dass die Nachricht verworfen wird, nachdem eine Berichtsnachricht zurückgegeben wurde. Wenn die Berichtsoptionen nicht Ihren Anforderungen entsprechen, können Sie eigene Sender schreiben, die andere `MQRO_EXCEPTION`- und `MQRO_DISCARD`-Berichtsoptionen verwenden. Wenn die SOAP-Anforderung

von einem anderen Sender gesendet wird, der MQRO_DISCARD nicht festgelegt hat, wird die fehlerhafte Nachricht in die Warteschlange für nicht zustellbare Nachrichten geschrieben.

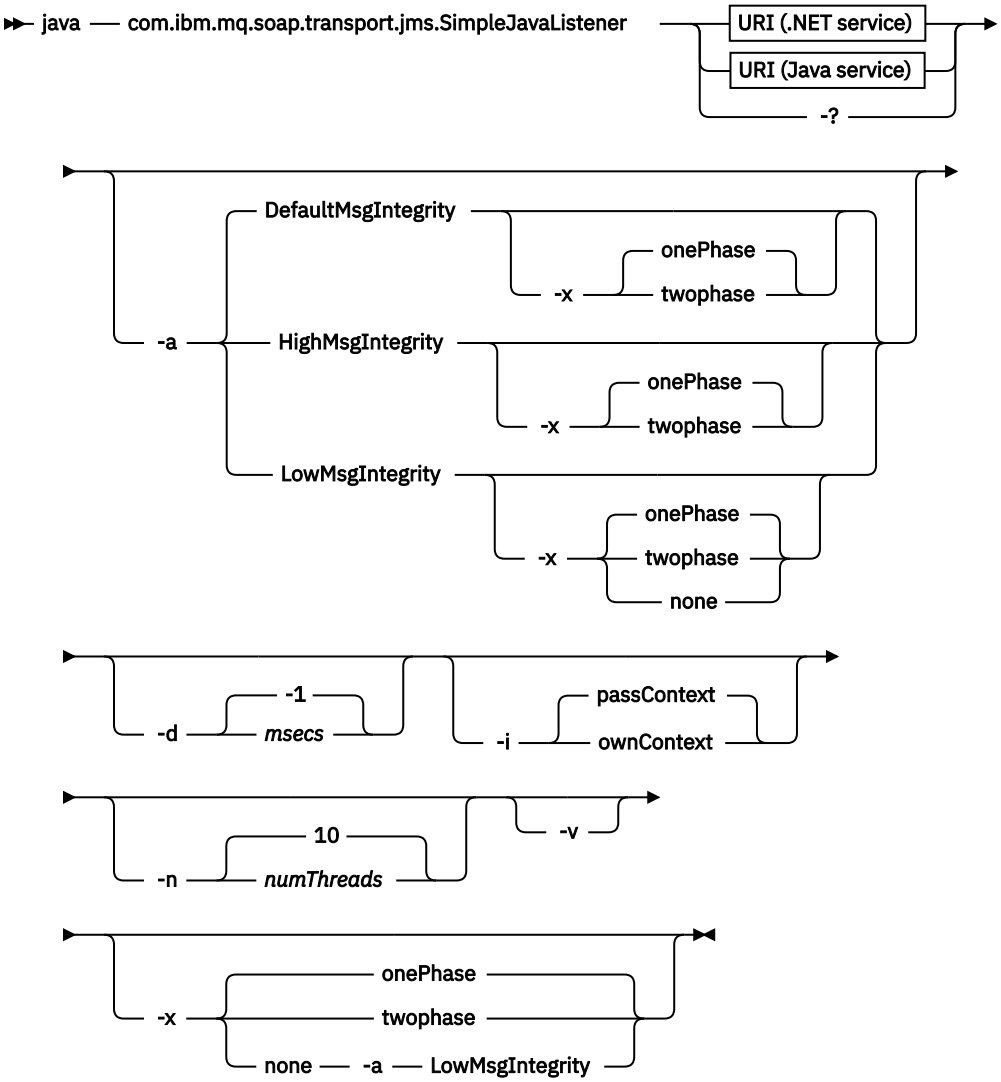
Wenn das Empfangsprogramm eine Berichtsnachricht generiert, jedoch beim Senden des Berichts ein Fehler auftritt, wird die Berichtsnachricht an die Warteschlange für nicht zustellbare Nachrichten gesendet. Stellen Sie sicher, dass Ihr Handler für die Warteschlange für nicht zustellbare Nachrichten diese Nachrichten ordnungsgemäß behandelt.

Wenn beim Versuch, in die Warteschlange für nicht zustellbare Nachrichten zu schreiben, ein Fehler auftritt, wird eine Nachricht in das Fehlerprotokoll von WebSphere MQ geschrieben. Ob das Empfangsprogramm anschließend mit dem Verarbeiten weiterer Nachrichten fortfährt, richtet sich danach, welche Nachrichtenpersistenz- und Transaktionsoptionen ausgewählt sind. Wenn das Empfangsprogramm im einphasigen Transaktionsmodus ausgeführt wird und eine nicht persistente Anforderungsnachricht verarbeitet, wird die ursprüngliche Nachricht verworfen. Das Empfangsprogramm von WebSphere MQ wird weiterhin ausgeführt. Wenn die Anforderungsnachricht persistent ist, wird sie in die Anforderungswarteschlange zurückgesetzt und das Empfangsprogramm wird beendet. Für die Anforderungswarteschlange ist "get-inhibited" festgelegt, um einen versehentlich ausgelösten Neustart zu verhindern.

Syntax diagram
.NET



Java



Erforderliche Parameter

URI ***platform***
Siehe „URI-Syntax und -Parameter für die Web-Service-Implementierung” auf Seite 1016.

-?
Hilfetext ausgeben, der beschreibt, wie der Befehl verwendet wird.

Optionale Parameter

-a ***integrityOption***
integrityOption gibt das Verhalten von SOAP-Empfangsprogrammen von WebSphere MQ an, wenn es nicht möglich ist, eine fehlgeschlagene Anforderungsnachricht in die Warteschlange für nicht zustellbare Nachrichten einzureihen. *integrityOption* kann einen der folgenden Werte haben:

DefaultMsgIntegrity
Bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung mit der ursprünglichen, verworfenen Nachricht fort. Bei persistenten Nachrichten zeigt es eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet. DefaultMsgIntegrity gilt, wenn die Option -a ausgelassen wird oder wenn *integrityOption* nicht angegeben ist.

LowMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Warnung an und setzt die Ausführung fort, wobei die Nachricht verworfen wird.

HighMsgIntegrity
Sowohl bei persistenten als auch bei nicht persistenten Nachrichten zeigt das Empfangsprogramm eine Fehlernachricht an, setzt die Anforderungsnachricht zurück, damit sie in der Anforderungswarteschlange bleibt, und wird beendet.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Wenn -x none angegeben ist, muss -a LowMsgIntegrity angegeben sein. Wenn die Flags nicht kompatibel sind, wird das Implementierungsdienstprogramm mit einer Fehlernachricht beendet, ohne dass irgendein Implementierungsschritt ausgeführt wurde.

-d ***msecs***
msecs gibt an, wie lange (in Millisekunden) das SOAP-Empfangsprogramm von WebSphere MQ aktiv bleibt, wenn in einem beliebigen Thread Nachrichten eingegangen sind. Wenn für *msecs* der Wert -1 festgelegt ist, bleibt das Empfangsprogramm für unbegrenzte Zeit aktiv.

-i ***Context***
Context gibt an, ob die Empfangsprogramme den Identitätskontext übergeben. *Context* kann die folgenden Werte haben:

passContext
Identitätskontext der ursprünglichen Anforderungsnachricht in die Antwortnachricht einfügen. Das SOAP-Empfangsprogramm prüft, ob es berechtigt ist, den Kontext aus der Anforderungswarteschlange zu speichern und an die Antwortwarteschlange zu übergeben. Es führt diese Prüfungen zur Laufzeit aus, wenn es die Anforderungswarteschlange zum Speichern des Kontextes und die Antwortwarteschlange zum Übergeben des Kontextes öffnet. Wenn es nicht über die erforderliche Berechtigung verfügt oder der MQOPEN-Aufruf fehlschlägt, wird die Antwortnachricht nicht verarbeitet. Die Antwortnachricht wird in die Warteschlange für nicht zustellbare Nachrichten eingereiht, wobei der Header der nicht zustellbaren Nachricht den Rückkehrcode aus dem fehlgeschlagenen MQOPEN-Aufruf enthält. Das Empfangsprogramm fährt anschließend mit der normalen Verarbeitung der nachfolgenden eingehenden Nachrichten fort.

ownContext
Das SOAP-Empfangsprogramm übergibt keinen Kontext. Der zurückgegebene Kontext gibt die Benutzer-ID an, unter der das Empfangsprogramm ausgeführt wird, und nicht die Benutzer-ID, die die ursprüngliche Anforderungsnachricht erstellt hat.

Die Felder im Ursprungskontext werden vom Warteschlangenmanager festgelegt, nicht vom SOAP-Empfangsprogramm.

-n ***numThreads***
numThreads gibt die Anzahl der Threads in den generierten Startscripts für das SOAP-Empfangsprogramm von WebSphere MQ an. Der Standardwert ist 10. Es kann sinnvoll sein, diesen Wert zu erhöhen, wenn der Nachrichtendurchsatz hoch ist.

-v
-v legt die ausführliche Ausgabe von externen Befehlen fest. Fehlernachrichten werden immer angezeigt. Verwenden Sie -v, um Befehle auszugeben, die Sie anpassen können, um angepasste Implementierungsscripts zu erstellen.

-w ***serviceDirectory***
serviceDirectory ist das Verzeichnis, das den Web-Service enthält.

-x ***transactionality***
transactionality gibt den Typ der Transaktionssteuerung für das Empfangsprogramm an. Für *transactionality* kann einer der folgenden Werte festgelegt sein:

onePhase
Es wird die Einphasenunterstützung von IBM WebSphere MQ verwendet. Wenn das System während der Verarbeitung ausfällt, wird die Anforderungsnachricht erneut der Anwendung zugestellt. WebSphere MQ -Transaktionen stellen sicher, dass die Antwortnachrichten genau einmal geschrieben werden.

twoPhase
Die Zweiphasenunterstützung wird verwendet. Wenn der Service entsprechend geschrieben wurde, wird die Nachricht genau einmal zugestellt, in Koordination mit anderen Ressourcen und innerhalb einer einzelnen festgeschriebenen Ausführung des Service. Diese Option gilt nur für Verbindungen mit Serverbindung.

none
Keine Transaktionsunterstützung. Wenn das System während der Verarbeitung ausfällt, kann die Anforderungsnachricht verloren gehen, auch wenn sie persistent ist. In diesem Fall ist es möglich, dass der Service ausgeführt wurde oder nicht und dass Antwort-, Berichts- oder nicht zustellbare Nachrichten geschrieben wurden oder nicht.

Das Implementierungsdienstprogramm prüft, ob die Flags -x und -a kompatibel sind. Weitere Informationen finden Sie in der Beschreibung zum Flag -a.

Beispiel für .NET

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

Java-Beispiel

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

Sender von IBM WebSphere MQ Transport for SOAP

Absenderklassen werden für Axis und .NET Framework 1 und .NET Framework 2 bereitgestellt. Der Sender erstellt eine SOAP-Anforderung und reiht sie in eine Warteschlange ein, bis er eine Antwort aus der Antwortwarteschlange gelesen hat. Sie können das Verhalten der Klassen ändern, indem Sie unterschiedliche URIs von einem SOAP-Client übergeben lassen. Für .NET Framework 3 verwenden Sie den angepassten WebSphere MQ-Kanal für Microsoft Windows Communication Foundation (WCF).

Verwendungszweck

Der SOAP-Sender von WebSphere MQ reiht eine SOAP-Anforderung zum Aufrufen eines Web-Service in eine WebSphere MQ-Anforderungswarteschlange ein. Der Sender legt im **MQRFH2**-Header Felder entsprechend den im URI angegebenen Optionen oder entsprechend Standardwerten fest.

Wenn Sie das Verhalten eines Senders in einer Weise ändern müssen, für die die URI-Optionen nicht ausreichen, müssen Sie einen eigenen Sender schreiben. Ihr Sender kann mit den Empfangsprogrammen von IBM WebSphere MQ Transport for SOAP oder mit anderen SOAP-Umgebungen zusammenarbeiten. Ihr Sender muss SOAP-Nachrichten in dem von WebSphere MQ definierten Format erstellen. Dieses Format wird vom SOAP-Listener von IBM WebSphere MQ sowie von den von WebSphere Application Server und CICS bereitgestellten SOAP-Listenern unterstützt. Ihr Sender muss die Regeln für einen IBM WebSphere MQ-Anforderer erfüllen. Das SOAP-Empfangsprogramm von IBM WebSphere MQ gibt Antwort- und Berichtsnachrichten zurück. Details zum Festlegen der Berichtsoptionen in der **MQMD**finden Sie unter „SOAP-Einstellungen für MQMD” auf Seite 992 . Die Berichtsoptionen steuern die Berichtsnachrichten, die vom SOAP-Empfangsprogramm von WebSphere MQ zurückgegeben werden.

Beschreibung

Der WebSphere MQ SOAP-Java-Sender wird in der Axis-Hostumgebung für das URI-Präfix `jms:` registriert. Der Sender ist in der Klasse `com.ibm.mq.soap.transport.jms.WMQSender` implementiert, die von `org.apache.axis.handlers.BasicHandler` abgeleitet ist. Wenn die Axis-Hostumgebung das URI-Präfix `jms:` erkennt, ruft sie die Klasse `com.ibm.mq.soap.transport.jms.WMQSender` auf. Die

Klasse wird nach dem Einreihen der Nachricht blockiert, bis eine Antwort aus der Antwortwarteschlange gelesen wurde. Wenn innerhalb eines Zeitlimitintervalls keine Antwort eingeht, löst der Sender eine Ausnahmebedingung aus. Wenn innerhalb des Zeitlimitintervalls eine Antwort eingeht, wird die Antwortnachricht mithilfe des Axis-Frameworks an den Client zurückgegeben. Ihre Clientanwendung muss in der Lage sein, mit diesen Antwortnachrichten umzugehen.

Für Microsoft .NET Framework 1- und .NET Framework 2-Services ist der SOAP-Sender von WebSphere MQ in der Klasse `IBM.WMQSOAP.MQWebRequest` implementiert, die von `System.Net.WebRequest` und `System.Net.IwebRequestCreate` abgeleitet ist. Wenn .NET Framework 1 oder .NET Framework 2 das URI-Präfix `jms:` erkennt, ruft es die Klasse `IBM.WMQSOAP.MQWebRequest` auf. Der Sender erstellt ein `MQWebResponse`-Objekt, um die Antwortnachricht aus der Antwortwarteschlange zu lesen und sie an den Client zurückzugeben.

`com.ibm.mq.soap.transport.jms.WMQSender` ist eine Endklasse, und `IBM.WMQSOAP.MQWebRequest` ist versiegelt. Sie können deren Verhalten nicht durch das Erstellen von Unterklassen ändern.

Parameter

Legen Sie den URI fest, um das Verhalten des SOAP-Senders von IBM WebSphere MQ im SOAP-Client eines Web-Service zu steuern. Das Implementierungsdienstprogramm erstellt Client-Stubs für den Web-Service anhand der URI-Optionen, die im Implementierungsdienstprogramm angegeben wurden.

Kanaldefinitionstabelle für den Sender von WebSphere MQ Transport for SOAP verwenden

Die Definition eines Clientverbindungskanals stellt eine Alternative zum Festlegen von Verbindungseigenschaften im Attribut ConnectionFactory des Web-Service-URI dar. Die Verbindungseigenschaften sind die Parameter clientChannel, clientConnection und SSL.

Beschreibung

Sie erstellen die Beschreibungstabelle für den Clientkanal, indem Sie Clientverbindungen definieren. Erstellen Sie alle Verbindungen in der Verbindungstabelle für einen einzelnen Warteschlangenmanager, auch wenn ein Web-Service-Client Verbindungen zu verschiedenen Warteschlangenmanagern herstellt. Der Standardname und die Standardposition der Verbindungstabelle lautet *queue manager directory*`/@ipcc/AMQCLCHL.TAB`.

Übergeben Sie die Position der Verbindungstabelle an einen Java-Client, indem Sie die Systemeigenschaft `com.ibm.mq.soap.transport.jms.mqchlurl` festlegen.

Übergeben Sie die Position der Verbindungstabelle an einen .NET-Client, indem Sie die Umgebungsvariablen `MQCHLLIB` und `MQCHLTAB` festlegen.

Im Attribut ConnectionFactory des Web-Service-URI können Sie sowohl eine Kanalverbindungstabelle als auch Kanalverbindungsparameter angeben. Die in ConnectionFactory festgelegten Werte haben Vorrang vor den Werten in der Kanaldefinitionstabelle.

Kanaldefinitionstabelle in Java verwenden

```
java -Dcom.ibm.msg.client.config.location=file:/C:/mydir/myjms.config MyAppClass
```

Abbildung 22. Java-Client mit einer Konfigurationsdatei starten

```
com.ibm.mq.soap.transport.jms.mqchlurl=file:/C:/ibm/wmq/qmgrs/QM1/@ipcc/AMQCLCHL.TAB
```

Abbildung 23. `myjms.config`

Transaktionen

Verwenden Sie beim Starten des Empfangsprogramms die Option `-x`, um Web-Services über Transaktionen auszuführen. Wählen Sie die Integrität von Nachricht aus, indem Sie die Option `persistence` im Service-URI festlegen.

Web-Services

Verwenden Sie beim Starten des Empfangsprogramms die Option `-x`, um Web-Services über Transaktionen auszuführen. Bei .NET Framework 1 und 2 verwendet das SOAP-Empfangsprogramm Microsoft Transaction Coordinator (MTS). Bei Axis 1.4 verwendet das SOAP-Empfangsprogramm vom Warteschlangenmanager koordinierte Transaktionen.

Web-Service-Clients

Die SOAP-Sender sind nicht transaktionsorientiert.

WebSphere MQ-Bindungen

Sie können den Bindungstyp für den SOAP-Sender festlegen. Er kann als WebSphere MQ-Serveranwendung oder als Clientanwendung eine Verbindung herstellen. Sie können den SOAP-Sender auch als XA-Client unter .NET binden.

Nachrichtenpersistenz

Wählen Sie die Ebene der Persistenz aus, indem Sie die Option `Persistence` im URI festlegen.

Web-Service-Transaktionen

Sie können Web-Service-Transaktionen verwenden, da der SOAP-Sender nicht transaktionsorientiert ist. Wenn Sie einen eigenen SOAP-Sender schreiben und Web-Service-Transaktionen verwenden möchten, erstellen Sie keinen transaktionsorientierten SOAP-Sender. Es ist nicht möglich, die Anforderungsnachricht und die Antwortnachricht in einer Transaktion zu senden. Der Sende- und Empfangsvorgang darf nicht von der Web-Service-Transaktion koordiniert werden.

URI-Syntax und -Parameter für die Web-Service-Implementierung

Die Syntax und die Parameter für die Implementierung eines IBM WebSphere MQ -Web-Service werden in einem URI definiert. Das Implementierungsdienstprogramm generiert einen Standard-URI auf der Grundlage des Namens des Web-Service. Sie können die Standardwerte überschreiben, indem Sie einen eigenen URI als Parameter für das Implementierungsdienstprogramm definieren. Das Implementierungsdienstprogramm fügt den URI in die generierten Client-Stubs für den Web-Service ein.

Verwendungszweck

Ein Web-Service wird mit einem Universal Resource Identifier (URI) angegeben. Dem Syntaxdiagramm können Sie den URI entnehmen, der in IBM WebSphere MQ Transport for SOAP unterstützt wird. Der URI steuert IBM WebSphere MQ-spezifische SOAP-Parameter und -Optionen für den Zugriff auf Zielservices. Der URI ist kompatibel mit Web-Services, die von .NET, Apache Axis 1, WebSphere Application Server, CICSgehostet werden.

Beschreibung

Der URI wird in die vom Implementierungsdienstprogramm generierten Clientklassen des Web-Service eingefügt. Der Client übergibt den URI in einer IBM WebSphere MQ-Nachricht an den SOAP-Sender von IBM WebSphere MQ. Der URI steuert die Verarbeitung sowohl des SOAP-Senders von IBM WebSphere MQ als auch des SOAP-Empfangsprogramms von IBM WebSphere MQ.

Syntax

The URI syntax is as follows:

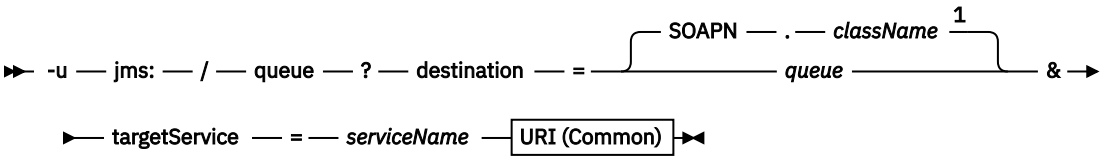
```
jms:/queue?name=value&name=value...
```

where name is a parameter name and *value* is an appropriate value, and the name=*value* element can be repeated any number of times with the second and subsequent occurrences being preceded by an ampersand (&).

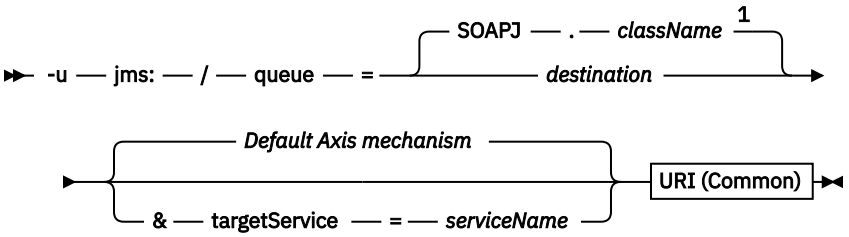
Parameter names are case-sensitive, as are names of IBM WebSphere MQ objects. If any parameter is specified more than once, the final occurrence of the parameter takes effect. Client applications can override a generated parameter by appending another copy of the parameter to the URI. If any additional unrecognized parameters are included, they are ignored.

If you store a URI in an XML string, you must represent the ampersand character as `&`. Similarly, if a URI is coded in a script, take care to escape characters such as & that would otherwise be interpreted by the shell.

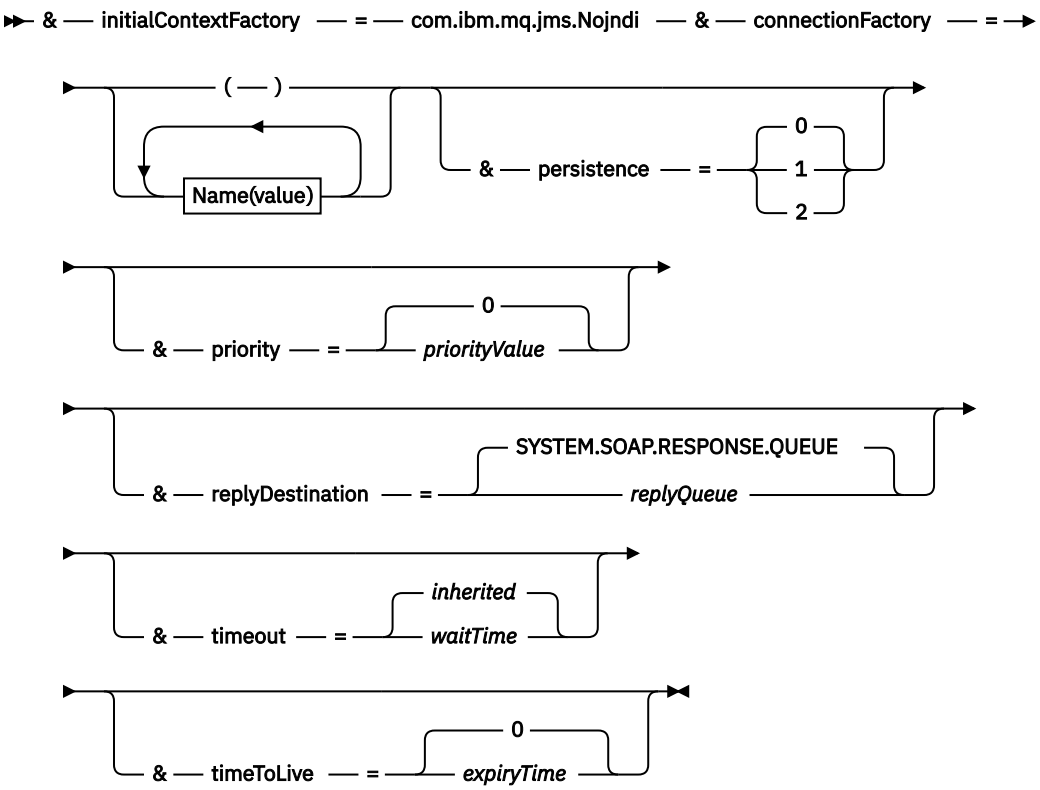
Syntax diagram
URI (.NET service)



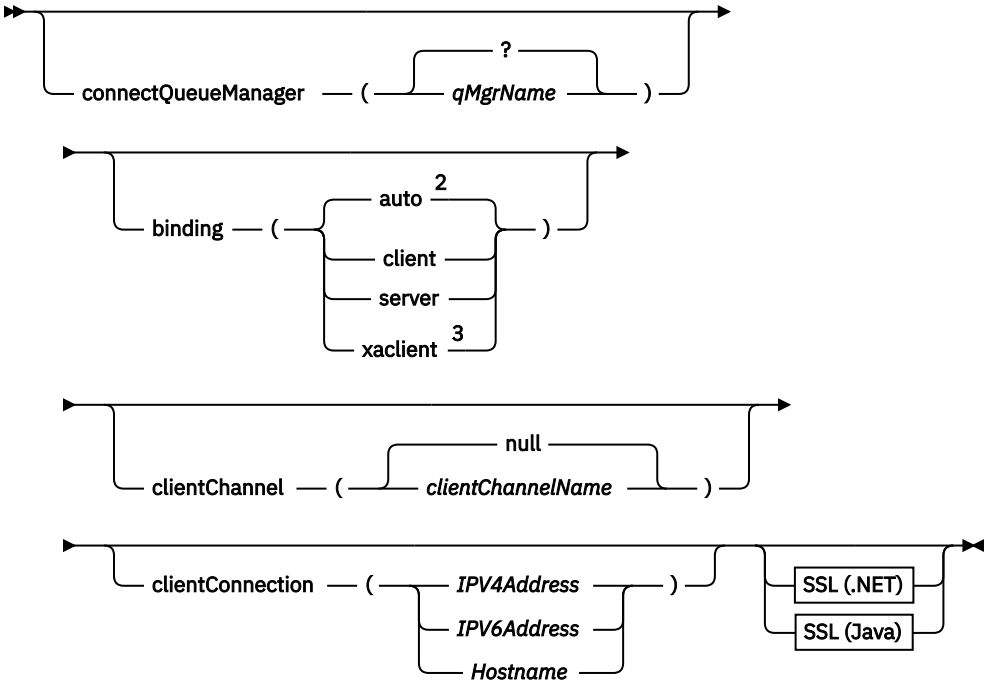
URI (Java service)



URI (Common)



***Name*(*value*)**



Anmerkungen:

[1] The queue manager transforms *className* to a queue name following the steps described in „Umwandlung des Ziels in einen Warteschlangennamen" auf Seite 1018

[2] `client` is the default if other options appropriate for a client are specified; for example `clientConnection`.

[3] `xaclient` applies to .NET only

Umwandlung des Ziels in einen Warteschlangennamen

1. *className* hat das Präfix `SOAPJ.` für Java -Services oder `SOAPN.` für .NET-Services.
2. Die Dateierweiterung wird aus dem vollständigen Pfadnamen entfernt, der im Parameter *className* angegeben ist.
3. Die sich daraus ergebende Zeichenfolge wird nach 48 Zeichen abgeschnitten.
4. Verzeichnistrennzeichen werden durch Punkte ersetzt.
5. Eingebettete Leerzeichen werden durch Unterstriche ersetzt.
6. Der Doppelpunkt nach einem Laufwerkbuchstaben wird bei .NET-Services durch einen Punkt ersetzt.

Anmerkung: In einigen Umgebungen kann es vorkommen, dass ein vom Implementierungsdienstprogramm generierter Warteschlangenname nicht eindeutig ist. Das Implementierungsdienstprogramm prüft, ob die Warteschlange erstellt werden soll. Auf Wunsch können Sie das Implementierungsdienstprogramm außer Kraft setzen, indem Sie die Implementierungsverzeichnishierarchie umstrukturieren oder den bereitgestellten Implementierungsprozess anpassen.

Erforderliche URI-Parameter

`destination`=*Warteschlange*

queue ist der Name des Anforderungsziels. Dabei kann es sich um eine Warteschlange oder um den Aliasnamen einer Warteschlange handeln. Wenn es der Aliasname einer Warteschlange ist, kann der Aliasname in ein Thema aufgelöst werden.

- Wenn der Parameter -u ausgelassen wird, wird *queue* aus *classname* generiert. Dafür werden die unter „Umwandlung des Ziels in einen Warteschlangennamen" auf Seite 1018 beschriebenen Schritte ausgeführt.

- Wenn der Parameter -u angegeben ist, ist *queue* erforderlich und muss der erste Parameter des URI nach der Zeichenfolge `jms:/queue?` am Anfang sein Zeichenfolge. Geben Sie entweder einen IBM WebSphere MQ -Warteschlangennamen oder einen Warteschlangennamen und einen Warteschlangenmanagernamen an, die durch ein @-Zeichen verbunden sind, z. B. `SOAPN.trandemos@WMQSOAP.DEMO.QM`.
- Das Implementierungsdienstprogramm prüft, ob der generierte oder angegebene Warteschlangenname mit dem Namen einer vorhandenen Warteschlange übereinstimmt. Die daraufhin getroffene Maßnahme wird in Tabelle 587 auf Seite 1019 beschrieben.

Tabelle 587. queue-Validierung

Empfangsprogrammscript vorhanden?	**Empfangsprogrammscript liegt im Verzeichnis `./generated/server` vor**		**Empfangsprogrammscript liegt nicht im Verzeichnis `./generated/server` vor**
Stimmt Warteschlange im Empfangsprogrammscript mit *queue* überein?	***queue* stimmt nicht mit der Anforderungswarteschlange überein, die im Empfangsprogrammscript verwendet wird**	***queue* stimmt mit Anforderungswarteschlange überein, die im Empfangsprogrammscript verwendet wird**	
***queue* ist vorhanden**	– *Implementierung wird mit Fehler beendet.* – Der Service wurde bereits unter `./generated/server` implementiert, jedoch mit einer anderen Warteschlange.	– **Implementierung wird regulär fortgesetzt.** – Der Service wurde bereits unter `./generated/server` implementiert.	– *Implementierung wird mit Fehler beendet.* – Das Startscript für das Empfangsprogramm wurde unter `./generated/server` nicht gefunden, jedoch wird *queue* von einem anderen Service oder einer anderen Anwendung verwendet.
***queue* ist nicht vorhanden**		– **Implementierung wird mit einer Warnung fortgesetzt.** – Die vorherige Implementierung ist möglicherweise fehlgeschlagen, denn der Start ist korrekt, doch *queue* fehlt.	– **Implementierung wird regulär fortgesetzt.** – In diesem Verzeichnis wurde bisher kein Service implementiert.

&connectionFactory=*Name (Wert)*
Name ist einer der folgenden Parameter:

- connectQueueManager (qMgrName)
- Bindung (bindingType)
- clientChannel(Kanal)
- clientConnection(Verbindung)
- „Erforderliche SSL-Parameter (Java)” auf Seite 1005

Eine Beschreibung der Werte dieser Parameter finden Sie unter „Parameter der Verbindungsfactory" auf Seite 1021.

&targetService=*serviceName*
[8]Unter .NET ist *serviceName* der Name eines .NET-Service, der sich im Implementierungsverzeichnis befindet, z. B. `targetService=myService.asmx`. In der .NET-Umgebung ermöglicht der Parameter targetService, dass ein einzelnes SOAP-Empfangsprogramm von WebSphere MQ Anforderungen für mehrere Services verarbeiten kann. Diese Services müssen im gleichen Verzeichnis implementiert sein.

Optionale URI-Parameter

&initialContextFactory=*contextFactory*
contextFactory ist erforderlich. Für diesen Parameter muss `com.ibm.mq.jms.Nojndi` festgelegt sein. Stellen Sie sicher, dass sich `Nojndi.jar` bei einem Web-Service-Client für WebSphere Application Server im Klassenpfad befindet. `Nojndi.jar` gibt Java-Objekte nicht durch Verweis auf ein Verzeichnis, sondern auf der Grundlage des Inhalts der Parameter `connectionFactory` und `destination` zurück.

&targetService=*serviceName*
[9]Auf Axis ist *serviceName* der vollständig qualifizierte Name eines Java -Service, z. B. `targetService=javaDemos.service.StockQuoteAxis`. Wenn targetService nicht angegeben ist, wird ein Service mit dem Axis-Standardmechanismus geladen.

&persistence=*messagePersistence*
messagePersistence hat einen der folgenden Werte:

0
Die Persistenz wird aus der Warteschlangendefinition übernommen.

1
Die Nachricht ist nicht persistent.

2
Die Nachricht ist persistent.

&priority=*priorityValue*
priorityValue liegt im Bereich von 0 bis 9. Dabei steht 0 für eine niedrige Priorität. Der Standardwert richtet sich nach der Umgebung. Bei IBM WebSphere MQ ist er 0.

&replyDestination=*replyTo*
Die Warteschlange auf Clientseite, die für die Antwortnachricht verwendet werden soll. Die Standardantwortschlange ist `SYSTEM.SOAP.RESPONSE.QUEUE`.

- Führen Sie das Script `setupWMQSOAP` aus, um die SOAP-Standardobjekte für WebSphere MQ zu erstellen.
- Geben Sie eine Modellwarteschlange für *replyToQueue* an, um entweder eine temporäre oder eine permanente dynamische Antwortwarteschlange zu erstellen. Sowohl bei temporären als auch bei permanenten dynamischen Antwortwarteschlangen wird für jede Anforderung eine separate Instanz der dynamischen Warteschlange erstellt. Wenn eines der folgenden Ereignisse eintritt, wird die Warteschlange gelöscht:
  - Die Antwort geht ein und wird verarbeitet.
  - Bei der Anforderung wird ein Zeitlimit überschritten.
  - Das anfordernde Programm wird beendet.

  Um die beste Leistung zu erzielen, sollten Sie temporäre dynamische Warteschlangen statt permanenter dynamischer Warteschlangen verwenden. Senden Sie keine persistente Anforderungsnachricht an einen URI mit einer temporären dynamischen Warteschlange. Das SOAP-Empfangspro-

[8] Nur .NET-Service
[9] Nur Java-Service

gramm von IBM WebSphere MQ kann in einem solchen Fall die Nachricht nicht verarbeiten und gibt einen Fehler aus. Der Client überschreitet das Zeitlimit, während er auf die Antwort wartet.

- Das Script `setupWMQSOAP` erstellt eine permanente dynamische Standardmodellwarteschlange namens `SYSTEM.SOAP.MODEL.RESPONSE.QUEUE`.

&timeout=*waitTime*
Gibt an, wie lange (in Millisekunden) der Client auf eine Antwortnachricht wartet. *waitTime* überschreibt die von der Infrastruktur oder der Clientanwendung festgelegten Werte. Wenn dieser Parameter nicht angegeben ist, wird der Anwendungswert (falls angegeben) oder der Standardwert der Infrastruktur übernommen.

Anmerkung: Zwischen "timeout" und timeToLive wird keine Beziehung erzwungen.

&timeToLive=*expiryTime*
expiryTime gibt an, nach wie vielen Millisekunden die Nachricht abläuft. Die Standardeinstellung ist null, d. h. die Laufzeit ist unbegrenzt.

Anmerkung: Zwischen "timeout" und "timeToLive" wird keine Beziehung erzwungen.

Parameter der Verbindungsfactory

connectQueueManager(*qMgrName*)
qMgrName gibt den Warteschlangenmanager an, zu dem der Client eine Verbindung herstellt. Der Standardwert ist leer.

binding(*bindingType*)
bindingType gibt an, wie der Client mit *qMgrName* verbunden ist. Die Standardeinstellung ist auto. *bindingType* kann die folgenden Werte haben:

auto
Der Sender testet nacheinander die folgenden Verbindungstypen:

1. Wenn andere, auf eine Clientverbindung anwendbare Optionen angegeben sind, verwendet der Sender eine Clientbindung. Die anderen Optionen sind clientConnection oder clientChannel.
2. Serververbindung verwenden.
3. Clientverbindung verwenden.

Verwenden Sie `binding(auto)` im *URI*, wenn es auf Seiten des SOAP-Clients keinen lokalen Warteschlangenmanager gibt. In diesem Fall wird für den SOAP-Client eine Clientverbindung hergestellt.

client
Verwenden Sie `binding(client)` im *URI*, um eine Clientkonfiguration für den SOAP-Sender zu erstellen.

server
Verwenden Sie `binding(server)` im *URI*, um eine Serverkonfiguration für den SOAP-Sender zu erstellen. Wenn für die Verbindung Parameter vom Typ "client" vorliegen, schlägt die Verbindung fehl und der SOAP-Sender von IBM WebSphere MQ zeigt einen Fehler an. Parameter vom Typ "client" sind clientConnection, clientChannel oder SSL-Parameter.

xaclient
xaclient ist nur auf .NET anwendbar, nicht auf Java-Clients. Es wird eine XA-Clientverbindung verwendet.

clientChannel(*Kanal*)

Der SOAP-Client verwendet den *Kanal* , um eine IBM WebSphere MQ -Clientverbindung herzustellen. *channel* muss mit dem Namen eines Serververbindungskanals übereinstimmen, es sei denn, beim Server ist die automatische Kanaldefinition aktiviert. clientChannel ist ein erforderlicher Parameter, sofern Sie keine Definitionstabelle für Clientverbindungen (Client Connection Definition Table, CCDT) bereitgestellt haben.
In Java stellen Sie eine CCDT bereit, indem Sie `com.ibm.mq.soap.transport.jms.mqchlurl` festlegen. In .NET legen Sie die Umgebungsvariablen `MQCHLLIB` und `MQCHLTAB` fest. Weitere

Informationen hierzu finden Sie unter „Kanaldefinitionstabelle für den Sender von WebSphere MQ Transport for SOAP verwenden” auf Seite 1015.

clientConnection(*Verbindung*)
Der SOAP-Client verwendet eine *Verbindung* , um eine IBM WebSphere MQ -Clientverbindung herzustellen.Der Standardhostname ist localhost und als Standardport wird 1414 verwendet. Wenn *connection* eine TCP/IP-Adresse ist, liegt diese in einem der folgenden drei Formate vor und eine Portnummer kann ihr nachgestellt sein.

JMS-Clients können entweder das Format: `hostname:port` verwenden oder die eckigen Klammern mit dem Format %X mit Escapezeichen versehen, wobei X der Hexadezimalwert ist, der die eckige Klammer in der Codepage des URI darstellt. In ASCII stehen %28 und %29 beispielsweise für `(` und `)`.

.Net-Clients können die eckigen Klammern explizit verwenden: `hostname(port)` oder das Format 'escaped' verwenden.

IPV4-Adresse
Beispiel: `192.0.2.0`

IPV6-Adresse
Beispiel: `2001:DB8:0:0:0:0:0:0`

Hostname
Beispiel: `www.example.com%281687%29`, `www.example.com:1687` oder `www.example.com(1687)`.

SSL *platform*
Informationen hierzu finden Sie unter „Erforderliche SSL-Parameter (Java)” auf Seite 1005.

Beispiel-URIs

Anmerkung:

1. & ist im URI als `&` codiert.
2. Alle zuvor aufgeführten Parameter sind auf die Clients anwendbar.
3. Nur **destination**, **connectionFactory** und **initialContextFactory** gelten für den WCF-Service.

```
jms:/queue?destination=myQ&amp;connectionFactory=()&amp;initialContextFacto
ry=com.ibm.mq.jms.Nojndi
```

Abbildung 24. URI für einen Axis-Service, in dem nur erforderliche Parameter angegeben werden

```
jms:/queue?destination=myQ&amp;connectionFactory=()&amp;targetService=MyService.asmx
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Abbildung 25. URI für einen .NET-Service, in dem nur erforderliche Parameter angegeben werden

```
jms:/queue?destination=myQ@myRQM&amp;connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Abbildung 26. URI für einen Axis-Service, in dem einige optionale connectionFactory-Parameter angegeben werden

```
jms:/queue?destination=myQ@myRQM&amp;connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Abbildung 27. URI für einen Axis-Service, in dem die Option sslPeerName des Parameters connectionFactory angegeben ist

Der Nojndi-Mechanismus

Der Nojndi-Mechanismus ermöglicht es JMS-Programmen, die JNDI-Schnittstellen verwenden, den gleichen URI wie WebSphere MQ-Programme zu nutzen, die JNDI nicht verwenden.

Sie können WebSphere MQ Transport for SOAP verwenden, um Web-Services auf WebSphere Application Server aufzurufen. Auf WebSphere Application Server fragt das SOAP over JMS-Protokoll die JMS-Ressourcen mittels JNDI ab. Der Web-Service-Client wird möglicherweise auf .NET ausgeführt oder verwendet Axis 1.4, um den Web-Service aufzurufen, und nutzt möglicherweise kein JNDI. Damit für den Client und den Server die gleiche URL verwendet wird, muss er die gleichen Informationen bereitstellen, ungeachtet dessen, ob die Umgebung JNDI verwendet oder nicht.

Der von einem Web-Service-Client an WebSphere MQ Transport for SOAP übergebene URI enthält einen bestimmten WebSphere MQ-Warteschlangenmanager und bestimmte Warteschlangennamen. Diese Namen werden analysiert und direkt von der WebSphere MQ-SOAP-Unterstützung verwendet.

Der Nojndi-Mechanismus leitet die von einem JMS-Programm verwendete initialContextFactory auf com.ibm.mq.jms.Nojndi um. Die Klasse com.ibm.mq.jms.Nojndi ist eine Implementierung der JNDI-Schnittstelle, die connectionFactory und destination aus der URL als ConnectionFactory -und Queue -Java-Objekte zurückgibt. Wenn es sich bei der JMS-Implementierung um WebSphere MQ handelt, übernehmen MQConnectionFactory und MQQueue ihre Werte von den Klassen ConnectionFactory und Queue.

Mit dem Nojndi-Mechanismus können Sie WebSphere Application Server und .NET die gleichen Verbindungsinformationen mit der gleichen URL bereitstellen.

SOAP over JMS-URI gemäß W3C für den WebSphere MQ-Axis-2-Client

Definieren Sie einen SOAP over JMS-URI gemäß W3C, um einen Web-Service von einem Axis 2-Client aus aufzurufen und dabei WebSphere MQ JMS als SOAP-Transportprotokoll zu verwenden. Der Web-Service muss von einem Server bereitgestellt werden, der WebSphere MQ JMS und den SOAP over JMS-Empfehlungskandidaten des W3C für die SOAP/JMS-Bindung unterstützt.

Beschreibung

Der W3C-Empfehlungskandidat (Candidate Recommendation) definiert die SOAP over JMS-Bindung; SOAP over Java Message Service 1.0. Ebenfalls nützlich aufgrund der enthaltenen Beispiele ist das Dokument URI Scheme for Java(tm) Message Service 1.0 .[10].

Verwenden Sie das Syntaxdiagramm, um SOAP over JMS-URIs gemäß W3C zu erstellen, die syntaktisch korrekt sind und vom WebSphere MQ-Axis-2-Client akzeptiert werden. Es beschränkt sich auf die Definition des URI, der vom WebSphere MQ-Axis-2-Client akzeptiert wird. Das Diagramm ist in zweierlei Hinsicht eine Untergruppe der W3C-Empfehlung:

1. Das Thema jms-variant wird nicht unterstützt und darf in einem URI, der an den WebSphere MQ-Axis-2-Client übergeben wird, nicht angegeben werden.
2. Die folgenden Eigenschaften sind im Syntaxdiagramm nicht enthalten, da es sich um JMS-Eigenschaften handelt und sie kein Teil des URI sind.
   a. bindingVersion
   b. contentType
   c. soapAction
   d. requestURI
   e. isFault

Die JMS-Eigenschaften werden vom Axis 2-Client oder vom Server festgelegt.

Das Diagramm erweitert die W3C-Empfehlung um einen benutzerdefinierten Parameter, connectionFactory. connectionFactory wird als Alternative zu JNDI verwendet, um anzugeben, wie der Axis 2-Client über eine Warteschlange eine Verbindung zu einem Warteschlangenmanager herstellt.

[10] Den aktuellen Entwurf finden Sie in den W3C-Spezifikationsreferenzen unter *URI Scheme for JMS*.

Der WebSphere MQ-Axis-2-Client akzeptiert Eigenschaften nur als Teil des URI, der von der Clientanwendung an den Client übergeben wird, oder als Umgebungsvariablen. Der WebSphere MQ-Axis-2-Client kann keine WSDL-Dokumente verarbeiten. Die Clientanwendung oder ein Entwicklungstool kann möglicherweise die WSDL verarbeiten und den an den Axis 2-Client zu übergebenden URI erstellen. Eine WebSphere MQ-Axis-2-Clientanwendung kann JMS-Nachrichteneigenschaften nicht direkt festlegen.

Syntax

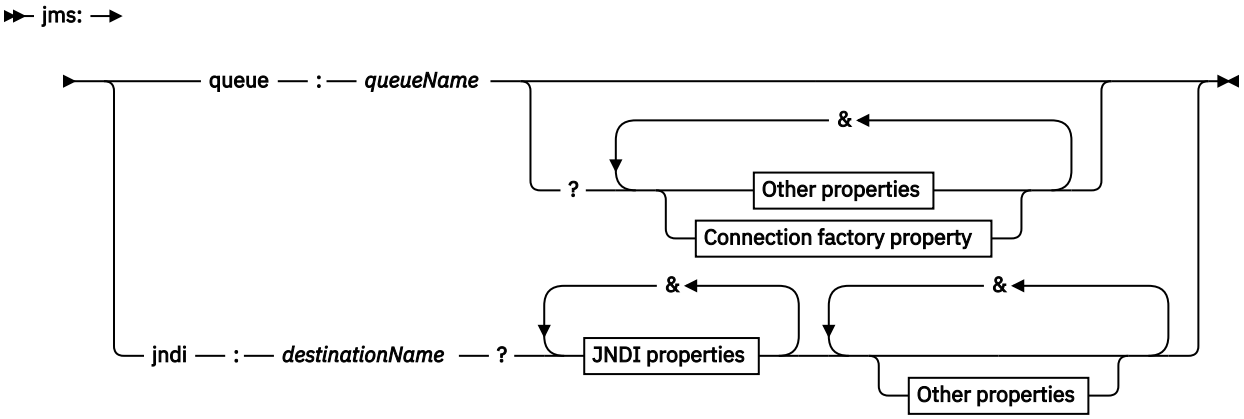
In accordance with the W3C recommendation, all the parameters can be obtained from environment variables. The environment variable names are formed by prefacing the parameter name with `soapjms_`. The syntax is: `soapjms_`*parameterName*; for example,

```
set soapjms_targetServer=com.example.org.stockquote
```

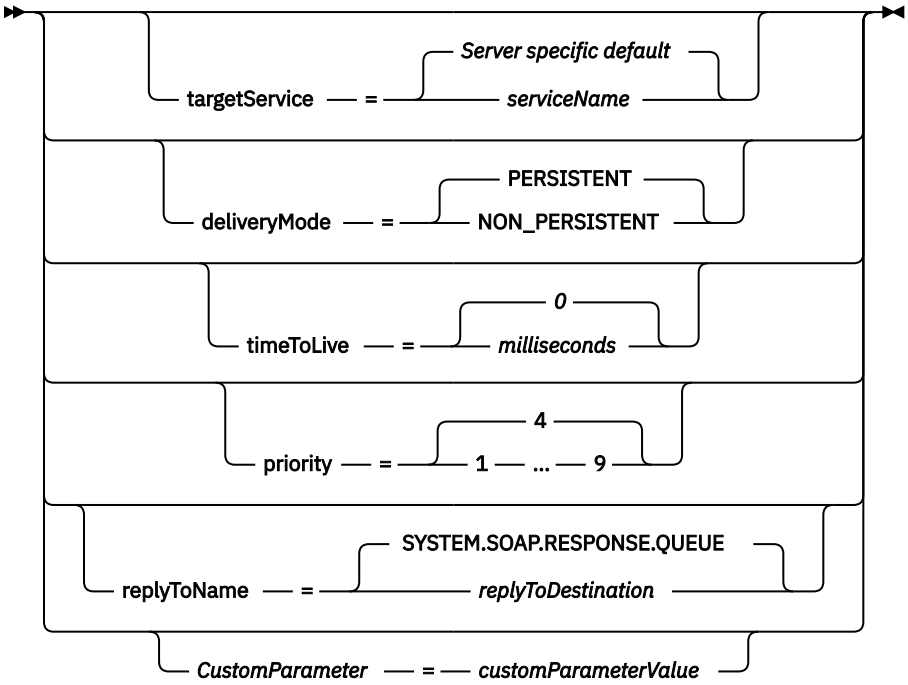
If a parameter is set using an environment variable it overrides the value set in the URI.

In accordance with the W3C recommendation, all the parameters can be repeated. The last instance of a parameter is used, unless overridden by an environment variable.

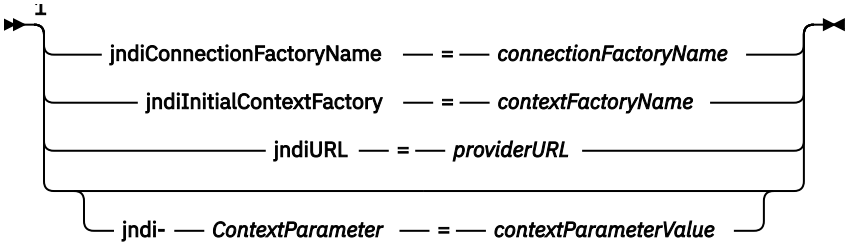
jms-uri



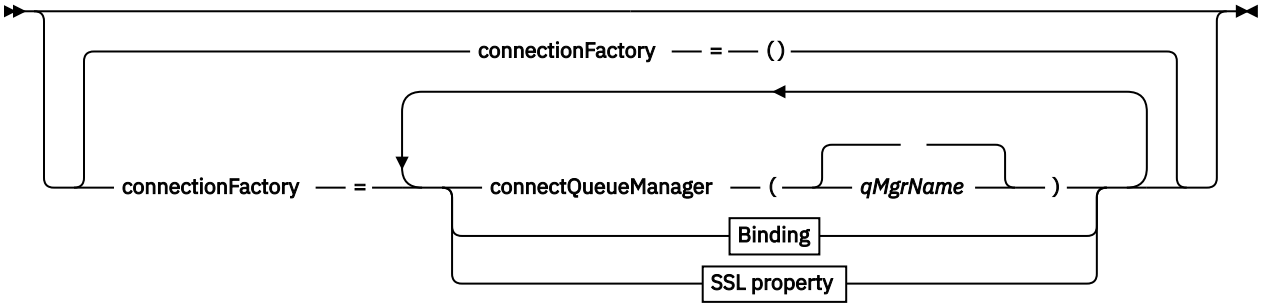
Other properties



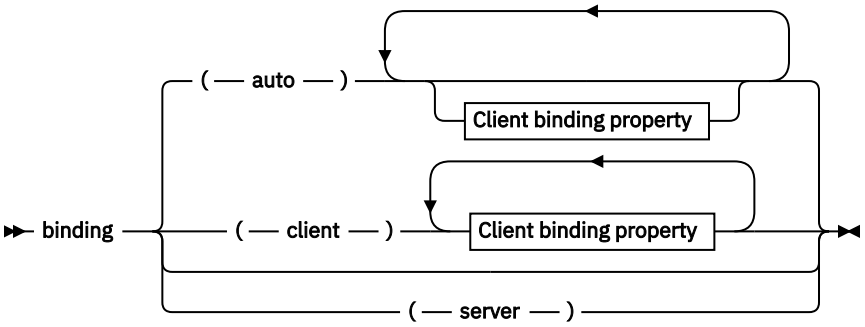
JNDI properties



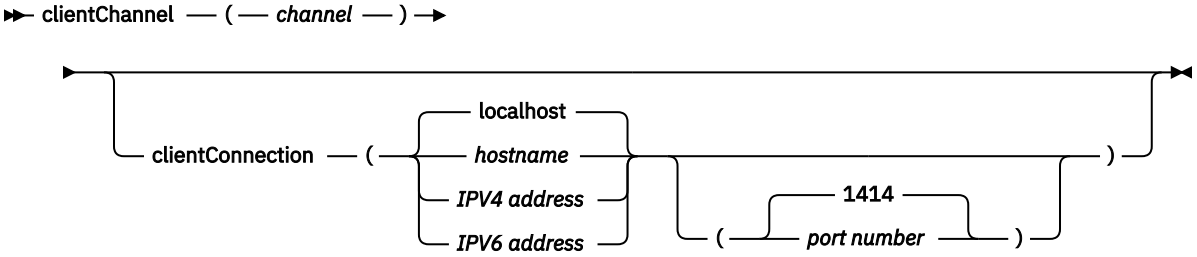
Connection factory property



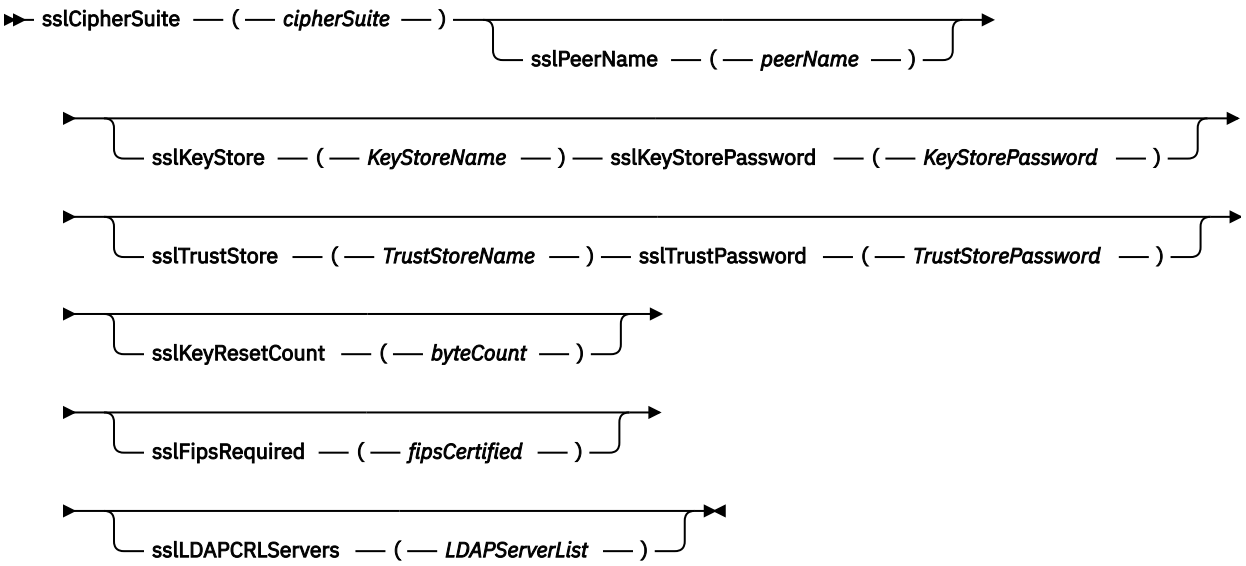
Binding



Client binding property



SSL property



Anmerkungen:

[1] **jndiConnectionFactoryName**, **jndiConnectionFactoryName** and **jndiURL** are all required parameters. **jndi-***ContextParameter* is optional.

Parameter

connectionFactory=*connectionFactoryParameterList*

connectionFactoryParameterList sind Parameter, die angeben, wie der Axis 2-Client eine Verbindung zu einem Warteschlangenmanager herstellt, wenn die Zielvariante queue ist.

connectionFactory darf nicht mit der Zielvariante jndi angegeben werden.

Die Parameter werden dem Server nicht im Anforderungs-URI übergeben.

Wenn connectionFactory ausgelassen wird, muss die Warteschlange zu einem Standard-Warteschlangenmanager gehören, der auf dem gleichen Server wie der Axis 2-Client ausgeführt wird.

Die *connectionFactoryParameterList*:

binding(*bindingType*)

bindingType gibt an, wie der Client mit *qMgrName* verbunden ist. Die Standardeinstellung ist auto. *bindingType* kann die folgenden Werte haben:

auto

Der Sender testet nacheinander die folgenden Verbindungstypen:

1. Wenn andere, auf eine Clientverbindung anwendbare Optionen angegeben sind, verwendet der Sender eine Clientbindung. Die anderen Optionen sind clientConnection oder clientChannel.
2. Serververbindung verwenden.
3. Clientverbindung verwenden.

Verwenden Sie `binding(auto)` im *URI*, wenn es auf Seiten des SOAP-Clients keinen lokalen Warteschlangenmanager gibt. In diesem Fall wird für den SOAP-Client eine Clientverbindung hergestellt.

client

Verwenden Sie `binding(client)` im *URI*, um eine Clientkonfiguration für den SOAP-Sender zu erstellen.

server

Verwenden Sie `binding(server)` im *URI*, um eine Serverkonfiguration für den SOAP-Sender zu erstellen. Wenn für die Verbindung Parameter vom Typ "client" vorliegen, schlägt die Ver-

bindung fehl und der SOAP-Sender von IBM WebSphere MQ zeigt einen Fehler an. Parameter vom Typ "client" sind clientConnection, clientChannel oder SSL-Parameter.

xaclient
xaclient ist nur auf .NET anwendbar, nicht auf Java-Clients. Es wird eine XA-Clientverbindung verwendet.

clientChannel(*Kanal*)

Der SOAP-Client verwendet den *Kanal* , um eine IBM WebSphere MQ -Clientverbindung herzustellen. *channel* muss mit dem Namen eines Serververbindungskanals übereinstimmen, es sei denn, beim Server ist die automatische Kanaldefinition aktiviert. clientChannel ist ein erforderlicher Parameter, sofern Sie keine Definitionstabelle für Clientverbindungen (Client Connection Definition Table, CCDT) bereitgestellt haben.

In Java stellen Sie eine CCDT bereit, indem Sie `com.ibm.mq.soap.transport.jms.mqchlurl` festlegen. In .NET legen Sie die Umgebungsvariablen `MQCHLLIB` und `MQCHLTAB` fest. Weitere Informationen hierzu finden Sie unter „Kanaldefinitionstabelle für den Sender von WebSphere MQ Transport for SOAP verwenden" auf Seite 1015.

clientConnection(*Verbindung*)
Der SOAP-Client verwendet eine *Verbindung* , um eine IBM WebSphere MQ -Clientverbindung herzustellen.Der Standardhostname ist localhost und als Standardport wird 1414 verwendet. Wenn *connection* eine TCP/IP-Adresse ist, liegt diese in einem der folgenden drei Formate vor und eine Portnummer kann ihr nachgestellt sein.

JMS-Clients können entweder das Format: `hostname:port` verwenden oder die eckigen Klammern mit dem Format %X mit Escapezeichen versehen, wobei X der Hexadezimalwert ist, der die eckige Klammer in der Codepage des URI darstellt. In ASCII stehen %28 und %29 beispielsweise für `(` und `)`.

.Net-Clients können die eckigen Klammern explizit verwenden: `hostname(port)` oder das Format 'escaped' verwenden.

IPV4-Adresse
Beispiel: `192.0.2.0`

IPV6-Adresse
Beispiel: `2001:DB8:0:0:0:0:0:0`

Hostname
Beispiel: `www.example.com%281687%29`, `www.example.com:1687` oder `www.example.com(1687)`.

sslCipherSuite(*CipherSuite*)
CipherSuite gibt die auf dem Kanal verwendete sslCipherSuite an. Die vom Client angegebene Cipher-Suite muss mit der auf dem Serververbindungskanal angegebenen Cipher-Suite übereinstimmen.

sslFipsRequired(*fipsCertified*)
fipsCertified gibt an, ob *CipherSpec* oder *CipherSuite* die FIPS-zertifizierte Verschlüsselung in IBM WebSphere MQ auf dem Kanal verwenden muss. Die Festlegung von *fipsCertified* hat dieselben Auswirkungen wie die Festlegung des Felds `FipsRequired` der **MQSCO**-Struktur in einem `MQCONNX`-Aufruf.

sslKeyStore(*KeyStoreName*)
KeyStoreName gibt den auf dem Kanal verwendeten sslKeyStoreName an. Im Keystore ist der private Schlüssel des Clients gespeichert, der verwendet wird, um den Client für den Server zu authentifizieren. Der Keystore ist optional, wenn die SSL-Verbindung so konfiguriert wurde, dass anonyme Clientverbindungen akzeptiert werden.

sslKeyResetCount(*bytecount*)
bytecount gibt die Anzahl der Byte an, die über einen SSL-Kanal übergeben werden, bevor der geheime SSL-Schlüssel neu vereinbart werden muss. Um die Neuvereinbarung von SSL-Schlüsseln zu inaktivieren, geben Sie für das Feld nichts oder null an. Null ist der einzige Wert, der in einigen Umgebungen unterstützt wird. Weitere Informationen finden Sie unter Geheime Schlüssel

in WebSphere MQ Classes for Java neu vereinbaren. Die Festlegung von sslKeyResetCount hat dieselben Auswirkungen wie die Festlegung des Felds KeyResetCount in der **MQSCO**-Struktur in einem MQCONNX-Aufruf.

sslKeyStorePassword(*KeyStorePassword*)
KeyStorePassword gibt das auf dem Kanal verwendete sslKeyStorePassword an.

sslLDAPCRLServers(*LDAPServerList*)

LDAPServerList gibt eine Liste mit LDAP-Servern an, die für die Überprüfung der Zertifikatswiderrufslisten verwendet werden sollen.

Bei SSL-fähigen Clientverbindungen ist *LDAPServerList* eine Liste mit LDAP-Servern, die für die Überprüfung der Zertifikatswiderrufslisten verwendet werden sollen. Das vom Warteschlangenmanager bereitgestellte Zertifikat wird mit einem der aufgelisteten LDAP-CRL-Servern abgeglichen; wenn es gefunden wird, schlägt die Verbindung fehl. Alle LDAP-Server werden der Reihe nach ausprobiert, bis eine funktionsfähige Verbindung mit einem Server hergestellt werden kann. Wenn mit keinem Server eine Verbindung hergestellt werden kann, wird das Zertifikat zurückgewiesen. Nach dem erfolgreichen Herstellen einer Verbindung mit einem Server wird das Zertifikat abhängig von den diesem LDAP-Server vorgelegten Zertifikatswiderrufslisten (CRLs, Certificate Revocation Lists) akzeptiert oder zurückgewiesen.

Wenn *LDAPServerList* leer ist, wird das Zertifikat für den Warteschlangenmanager nicht mit einer Zertifikatswiderrufsliste abgeglichen. Es wird eine Fehlernachricht angezeigt, wenn die bereitgestellte Liste mit LDAP-URIs ungültig ist. Die Festlegung dieses Felds hat dieselben Auswirkungen wie die Einbeziehung von MQAIR-Datensätzen und der Zugriff auf diese über eine **MQSCO**-Struktur in einem MQCONNX-Aufruf.

sslPeerName(*peerName*)
peerName gibt den auf dem Kanal verwendeten sslPeerName an.

sslTrustStore(*TrustStoreName*)
TrustStoreName gibt den auf dem Kanal verwendeten sslTrustStoreName an. Im Truststore ist das öffentliche Zertifikat des Servers oder die zugehörige Schlüsselkette gespeichert, die verwendet wird, um den Server für den Client zu authentifizieren. Der Truststore ist optional, wenn das Stammzertifikat einer Zertifizierungsstelle zum Authentifizieren des Servers verwendet wird. In Java werden Stammzertifikate im JRE-Zertifikatsspeicher cacerts gespeichert.

sslTrustStorePassword(*TrustStorePassword*)
TrustStorePassword gibt das auf dem Kanal gespeicherte sslTrustStorePassword an.

CustomParameter=customParameterValue

CustomParameter ist der benutzerdefinierte Name eines benutzerdefinierten Parameters und *customParameterValue* ist der Wert des Parameters.

Benutzerdefinierte Parameter, die nicht vom Axis 2-Client verwendet werden, werden vom Axis 2-Client an den SOAP-Server gesendet. Weitere Informationen finden Sie in der Dokumentation zum Server. connectionFactory ist ein benutzerdefinierter Parameter, der vom Axis 2-Client verwendet wird und nicht an den Server übergeben wird.

CustomParameter darf nicht mit dem Namen eines vorhandenen Parameters übereinstimmen.

Wenn *CustomParameter* mit der Zeichenfolge jndi- beginnt, wird er beim Suchen eines JNDI-Ziels verwendet (siehe jndi-).

deliveryMode=*deliveryMode*
deliveryMode legt die Nachrichtenpersistenz fest. Der Standardwert ist PERSISTENT.

jndi:*destinationName*
destinationName ist der Name eines JNDI-Ziels, der einer JMS-Warteschlange zugeordnet ist. Wenn die Zielvariante jndi angegeben ist, müssen Sie einen *destinationName* angeben.

jndiConnectionFactoryName=*connectionFactoryName*
connectionFactoryName legt den JNDI-Namen der Verbindungsfactory fest. Wenn die Zielvariante jndi ist, muss *connectionFactoryName* angegeben werden.

jndiInitialContextFactory=*contextFactoryName*
contextFactoryName legt den JNDI-Namen der Ausgangskontextfactory fest. Wenn die Zielvariante jndi ist, muss *contextFactoryName* angegeben werden. Weitere Informationen finden Sie unter JNDI zum Abrufen verwalteter Objekte in einer JMS-Anwendung verwenden.

jndiURL=*providerURL*
jndiURL legt den URL-Namen des JNDI-Providers fest. Wenn die Zielvariante jndi ist, muss *jndiURL* angegeben werden.

jndi-*ContextParameter*=*contextParameterValue*
jndi-*ContextParameter* ist der benutzerdefinierte Name eines benutzerdefinierten Parameters, mit dem Informationen an den JNDI-Provider übergeben werden. *contextParameterValue* ist die Information, die übergeben wird.

priority=*priorityValue*
priorityValue legt die JMS-Nachrichtenpriorität fest. Der Wert `0` steht für eine niedrige, 9 für eine hohe Priorität. Der Standardwert ist 4.

queue:*queueName*
queueName ist der Name einer JMS-Warteschlange, in die die SOAP-Anforderung eingereiht wird. Wenn die Warteschlangenvariante angegeben ist, muss ein Warteschlangenname angegeben werden. Wenn die Warteschlange zu keinem Standard-Warteschlangenmanager auf dem gleichen Server wie der Client gehört, legen Sie den Parameter connectionFactory fest.

replyToName=*replyToDestination*
replyToDestination legt den Namen der Zielwarteschlange fest. Wenn die Zielvariante jndi ist, handelt es sich bei dem Namen um einen JNDI-Namen, der einer Warteschlange zugeordnet sein muss. Wenn die Variante queue ist, ist der Name eine JMS-Warteschlange. Der Standardwert ist `SYSTEM.SOAP.RESPONSE.QUEUE`.

targetService=*serviceName*
Der Name, der vom SOAP-Server verwendet wird, um den Ziel-Web-Service zu starten.

> Auf Axis ist *serviceName* der vollständig qualifizierte Name eines Java-Service, z. B. `targetService=www.example.org.StockQuote`. Wenn targetService nicht angegeben ist, wird ein Service mit dem Axis-Standardmechanismus geladen.

timeToLive=*milliseconds*
Legen Sie für *milliseconds* die Frist bis zum Ablaufen der Nachricht fest. Die Standardeinstellung `0` bedeutet, dass die Nachricht nie abläuft.

Beispiele

```
jms:jndi:REQUESTQ
    ?jndiURL=file:/C:/JMSAdmin
    &jndiInitialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory
    &jndiConnectionFactoryName=ConnectionFactory
    &replyToName=RESPONSEQ
    &deliveryMode(NON_PERSISTENT)
```

Abbildung 28. Verwendung von jms:jndi zum Senden einer SOAP-/JMS-Anforderung

```
jms:queue:SOAPJ.demos
    ?connectionFactory=connectQueueManager(QM1)
     Bind(Client)
     ClientChannel(SOAPClient)
     ClientConnection(www.example.org(1418))
    &deliveryMode(NON_PERSISTENT)
```

Abbildung 29. Verwendung von jms:queue zum Senden einer SOAP-/JMS-Anforderung

Unterstützte Web-Services

Code, der zur Ausführung als Web-Service geschrieben wurde, muss nicht geändert werden, um IBM WebSphere MQ Transport for SOAP verwenden zu können. Services müssen für die Ausführung mit IBM WebSphere MQ Transport for SOAP anders implementiert werden als bei Verwendung von HTTP.

Beschreibung

WebSphere MQ Transport for SOAP stellt für die Ausführung von Services für .NET Framework 1, .NET 2 und Axis 1.4 ein SOAP-Empfangsprogramm bereit. Der angepasste Kanal WebSphere MQ für Microsoft Windows Communication Foundation führt Services für .NET Framework 3 aus. WebSphere Application Server und CICS unterstützen die Ausführung von Services über WebSphere MQ Transport for SOAP. Erstellen Sie einen benutzerdefinierten Export, um WebSphere Enterprise Service Bus oder WebSphere Process Server zu verwenden.

Das SOAP-Empfangsprogramm von WebSphere MQ kann SOAP-Anforderungen als Transaktionen verarbeiten. Führen Sie **amqwdeployWMQService** mit der Option -x aus. Die Zweiphasenoption wird nur für Empfangsprogramme unterstützt, die Serverbindungen verwenden. Andere Umgebungen stellen möglicherweise Transaktionsunterstützung für WebSphere MQ Transport for SOAP bereit. Lesen Sie dafür in der Dokumentation zur Umgebung nach.

WebSphere MQ Transport for SOAP unterstützt derzeit nicht das SOAP over JMS-Protokoll, das beim W3C eingereicht wurde und sich zum Industriestandard entwickelt. Eine SOAP-/JMS-Nachricht, die für den neuen Standard geschrieben wurde, ist an der JMS-Eigenschaft BindingVersion zu erkennen. WebSphere MQ Transport for SOAP legt die Eigenschaft BindingVersion nicht fest.

Axis 1.4

Eine Java-Klasse kann normalerweise unverändert verwendet werden. Die Argumenttypen für die Methoden im Web-Service müssen von der Axis-Engine unterstützt werden. Weitere Informationen finden Sie in der Axis-Dokumentation. Wenn der Service ein komplexes Objekt als Argument verwendet oder zurückgibt, muss dieses Objekt der Java™-Spezifikation entsprechen. Beispiele hierzu finden Sie in Abbildung 32 auf Seite 1032, Abbildung 33 auf Seite 1032 und Abbildung 34 auf Seite 1032:

1. Muss über einen öffentlichen, parameterlosen Konstruktor verfügen.
2. Komplexe Typen des Beans müssen über öffentliche Getter und Setter der folgenden Form verfügen:

Bereiten Sie den Service mithilfe des Dienstprogramms **amqwdeployWMQService** auf die Implementierung vor. Der Service wird vom SOAP-Empfangsprogramm von WebSphere MQ aufgerufen, welches `axis.jar` zum Ausführen des Service verwendet.

Der einzige zweiphasige Transaktionsmanager, der für Axis 1.4 unterstützt wird, ist WebSphere MQ.

Das bereitgestellte Implementierungsdienstprogramm unterstützt nicht den Fall, dass ein Service ein Objekt in einem anderen Paket an den Service selbst zurückgibt. Um ein Objekt verwenden zu können, das in einem anderen Paket zurückgegeben wird, müssen Sie Ihr eigenes Implementierungsdienstprogramm schreiben. Sie können das bereitgestellte Muster als Grundlage für Ihr Implementierungsdienstprogramm verwenden oder die von ihm abgesetzten Befehle mit der Option -v erfassen. Ändern Sie die Befehle wie erforderlich, um ein angepasstes Script zu erstellen.

Wenn der Service Klassen verwendet, die nicht zur Axis-Infrastruktur und zur SOAP-Laufzeitumgebung von WebSphere MQ gehören, müssen Sie den richtigen CLASSPATH festlegen. Um den CLASSPATH zu ändern, ändern Sie das generierte Script, das die Empfangsprogramme startet oder definiert, um die benötigten Services einzubeziehen. Dafür haben Sie die folgenden Möglichkeiten:

- Ändern Sie den CLASSPATH direkt im Script nach dem Aufruf von **amqwsetcp**.
- Erstellen Sie ein servicespezifisches Script zum Anpassen des CLASSPATH und rufen Sie dieses Script im generierten Skript nach dem Aufruf von **amqwsetcp** auf.
- Erstellen Sie einen angepassten Implementierungsprozess, um den CLASSPATH im generierten Script automatisch anzupassen.

.NET Framework 1 und .NET Framework 2

Ein Service, der bereits als HTTP-Web-Service erstellt wurde, muss nicht geändert werden, um als Web-Service für WebSphere MQ verwendet werden zu können. Er muss mithilfe des Dienstprogramms **amqwdeployWMQService** implementiert werden.

Der einzige zweiphasige Transaktionsmanager, der für .NET Framework 1 und .NET 2 unterstützt wird, ist Microsoft Transaction Server (MTS).

Wenn der Service-Code nicht als HTTP-Web-Service erstellt wurde, muss er in einen Web-Service umgewandelt werden. Deklarieren Sie die Klasse als Web-Service und geben Sie an, wie die Parameter der einzelnen Methoden formatiert sind. Sie müssen sicherstellen, dass alle Argumente für die Methoden des Service mit der Umgebung kompatibel sind.Abbildung 30 auf Seite 1032 und Abbildung 31 auf Seite 1032 zeigen eine .NET-Klasse, die als Web-Service vorbereitet wurde. Die vorgenommenen Ergänzungen sind fett dargestellt.

Abbildung 30 auf Seite 1032 verwendet das Code-Behind-Programmiermodell für .NET-Web-Services. Beim Code-Behind-Modell ist der Quellcode für den Service von der Datei `.asmx` getrennt. Die Datei `.asmx` deklariert den Namen der zugehörigen Quellendatei mit dem Schlüsselwort Codebehind. In WebSphere MQ stehen Beispiele sowohl für Inline- als auch für Code-Behind-Web-Services für .NET bereit.

Der Quellcode von .NET-Web-Services muss vor der Implementierung mit dem Implementierungsdienstprogramm **amqwdeployWMQService** kompiliert werden. Der Service wird in eine Bibliothek kompiliert (`.dll`). Die Bibliothek muss sich im Unterverzeichnis `./bin` des Implementierungsverzeichnisses befinden.

.NET Framework 3

Erstellen Sie einen angepassten WebSphere MQ -Kanal für Microsoft Windows Communication Foundation (WCF), um Services aufzurufen, die in .NET Framework 3 bereitgestellt werden. Im Abschnitt IBM WebSphere MQ Custom Channel for Microsoft Windows Communication Foundation (WCF) finden Sie eine Beschreibung der Konfiguration von WCF für die Verwendung des WebSphere MQ -Transports für SOAP.

WebSphere Application Server

Sie können Web-Services, die von WebSphere Application Server gehostet werden, mit WebSphere MQ Transport for SOAP aufrufen. Weitere Informationen finden Sie im Abschnitt SOAP over JMS für den Transport von Web-Services verwenden (veraltet).

Sie müssen die WSDL ändern, die bei der Implementierung eines JMS-Service für WebSphere Application Server generiert wurde, um einen Web-Service-Client zu generieren. Die bei der Implementierung für WebSphere Application Server erstellte WSDL enthält einen URI mit einer JNDI-Referenz auf die `InitialContextFactory` von JMS. Sie müssen die JNDI-Referenz so ändern, dass sie auf Nojndi verweist, und wie unter „URI-Syntax und -Parameter für die Web-Service-Implementierung” auf Seite 1016 beschrieben Verbindungsattribute angeben.

CICS

CICS-Anwendungen können mit WebSphere MQ Transport for SOAP aufgerufen werden (siehe CICS-System für Web-Services konfigurieren.

WebSphere Enterprise Service Bus und WebSphere Process Server for Multiplatforms

WebSphere ESB und WebSphere Process Server for Multiplatforms unterstützen SOAP over JMS mit einer vorgefertigten Bindung, jedoch nur, wenn der standardmäßige Messaging-Provider von WebSphere Application Server verwendet wird. Erstellen Sie eine benutzerdefinierte Bindung für JMS, damit WebSphere MQ Transport for SOAP unterstützt wird. Siehe JMS-Datenbindungen.

Beispiel

```
<%@ WebService Language="C#" CodeBehind="Quote.asmx.cs" Class="Quote.QuoteDotNet" %>
```

Abbildung 30. Servicedefinition für .NET Framework 2: `Quote.asmx`

```
<%@ WebService Language="C#" CodeBehind="Quote.asmx.cs" Class="Quote.QuoteDotNet" %>
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

namespace Quote {
    [WebService(Namespace = "http://www.example.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class QuoteDotNet : System.Web.Services.WebService {
        [WebMethod]
        public string getQuote(String symbol){
            return symbol.ToUpper();
        }
    }
}
```

Abbildung 31. Serviceimplementierung für .NET Framework 2: `Quote.asmx.cs`

```
package org.example.www;
public interface CustomerInfoInterface extends java.rmi.Remote {
    public org.example.www.CustomerRecord
           getCustomerName(org.example.www.CustomerRecord request)
           throws java.rmi.RemoteException, org.example.www.GetCustomerName_faultMsg;
}
```

Abbildung 32. Java-JAX-RPC-Serviceschnittstelle mit komplexem Typ

```
package org.example.www;
public class CustomerInfoPortImpl implements org.example.www.CustomerInfoInterface{
    public org.example.www.CustomerRecord
           getCustomerName(org.example.www.CustomerRecord request)
           throws java.rmi.RemoteException, org.example.www.GetCustomerName_faultMsg {
        request.setName(request.getID().toString());
        return request;
    }
}
```

Abbildung 33. Java-JAX-RPC-Serviceimplementierung mit komplexem Typ

```
package org.example.www;
public class CustomerRecord  {
    private java.lang.String name;
    private java.lang.Integer ID;
    public CustomerRecord() {}
    public java.lang.String getName() {
        return name; }
    public void setName(java.lang.String name) {
        this.name = name; }
    public java.lang.Integer getID() {
        return ID; }
    public void setID(java.lang.Integer ID) {
        this.ID = ID; }
}
```

Abbildung 34. Java-JAX-RPC-Service-Bean-Implementierung eines komplexen Typs

Web-Service-Client für IBM WebSphere MQ Transport for SOAP

Sie können einen vorhandenen Client für SOAP over HTTP für IBM WebSphere MQ Transport for SOAP wiederverwenden. Sie einige geringfügige Änderungen am Code und am Erstellungsprozess vornehmen, um den Client zur Verwendung mit IBM WebSphere MQ Transport for SOAP entsprechend anzupassen.

Codierung

JAX-RPC-Clients müssen in Java erstellt werden; .NET Framework 1- und .NET Framework 2-Clients können mit jeder Programmiersprache erstellt werden, die Common Language Runtime verwendet. Im Folgenden finden Sie Codebeispiele in C# und Visual Basic.

Das Ausmaß der Transaktionsunterstützung richtet sich nach der Clientumgebung und nach dem Muster der SOAP-Interaktion. Die SOAP-Anforderung und die SOAP-Antwort dürfen nicht Teil der gleichen atomaren Transaktion sein.

Auf einem .NET Framework 1- oder .NET Framework 2-Client müssen Sie `IBM.WMQSOAP.Register.Extension()` aufrufen. In einem Java-Web-Service auf der Grundlage von JAX-RPC müssen Sie `com.ibm.mq.soap.Register.extension` aufrufen, um den SOAP-Sender von WebSphere MQ SOAP zu registrieren. Die Methode registriert den Sender von WebSphere MQ Transport for SOAP als Handler für SOAP-Nachrichten bei Verwendung des `jms:`-Protokolls.

Um ein .NET Framework 3-Client zu erstellen, generieren Sie einen Windows Communication Foundation-Client-Proxy mit dem Tool **svcutil**. Weitere Informationen hierzu finden Sie unter WCF-Client-Proxy und Anwendungskonfigurationsdateien mit dem Tool "svcutil" mit Metadaten aus einem aktiven Service erstellen.

Zum Erstellen und Ausführen von .NET Framework 1- und .NET Framework 2-Clients erforderliche Bibliotheken

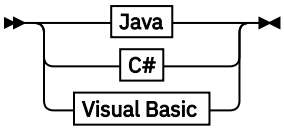
- amqsoap
- System
- System.Web.Services
- System.Xml

Zum Erstellen und Ausführen von Axis 1.4-Clients erforderliche Bibliotheken

- *MQ_Install*\java\lib\com.ibm.mq.soap.jar;
- *MQ_Install*\java\lib\com.ibm.mq.commonservices.jar;
- *MQ_Install*\java\lib\soap\axis.jar;
- *MQ_Install*\java\lib\soap\jaxrpc.jar
- *MQ_Install*\java\lib\soap\saaj.jar;
- *MQ_Install*\java\lib\soap\commons-logging-1.0.4.jar;
- *MQ_Install*\java\lib\soap\commons-discovery-0.2.jar;
- *MQ_Install*\java\lib\soap\wsdl4j-1.5.1.jar;
- *MQ_Install*\java\jre\lib\xml.jar;
- *MQ_Install*\java\lib\soap\servlet.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jar;
- *MQ_Install*\java\lib\com.ibm.mq.headers.jar;
- *MQ_Install*\java\lib\com.ibm.mq.pcf.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.remote.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.local.jar;

- *MQ_Install*\java\lib\connector.jar;
- *MQ_Install*\java\lib\jta.jar;
- *MQ_Install*\java\lib\jndi.jar;
- *MQ_Install*\java\lib\ldap.jar

Register SOAP extension



Java

C#

Visual Basic

Clientbeispiele

Abbildung 35 auf Seite 1034 ist ein Beispiel für einen in C# geschriebenen .NET Framework 1- oder .NET Framework 2-Client, der das Inline-Programmiermodell verwendet. Die Methode **IBM.WMQSOAP.Register.Extension()** registriert den SOAP-Sender von WebSphere MQ für .NET als `jms:`-Protokollhandler.

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
             }  catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
             }
        }
    }
}
```

Abbildung 35. Beispiel: Web-Service-Client in C#

Abbildung 36 auf Seite 1035 ist ein Beispiel für einen Java-Client, der die statische Proxy-Client-Schnittstelle JAX-RPC verwendet. Die Methode **com.ibm.mq.soap.Register.extension();** registriert den SOAP-Sender von WebSphere MQ für den Service-Proxy als Handler für das `jms:`-Protokoll.

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                  + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Abbildung 36. Beispiel: Web-Service-Client in Java

Referenzinformationen zu Benutzerexits, API-Exits und installierbaren Services

Verwenden Sie die in diesem Abschnitt bereitgestellten Links als Unterstützung bei der Entwicklung Ihrer Anwendungen für Benutzerexits, API-Exits und installierbare Services:

- „MQIEP-Struktur” auf Seite 1036
- „Datenkonvertierungsexit-Referenz” auf Seite 1039
- „Veröffentlichungsexit - MQ_PUBLISH_EXIT” auf Seite 1043
- „Kanalexitaufrufe und Datenstrukturen” auf Seite 1051
- „API-Exitreferenz” auf Seite 1116
- „Referenzinformationen zu installierbaren Services” auf Seite 1177

Zugehörige Konzepte

„Referenzinformationen zu MQI-Anwendungen” auf Seite 7
Verwenden Sie die Informationen der Links in diesem Kapitel beim Entwickeln Ihrer MQI-Anwendungen:

„IBM WebSphere MQ -Klassen für Java-Bibliotheken” auf Seite 1458
Die Position der IBM WebSphere MQ -Klassen für Java-Bibliotheken variiert je nach Plattform. Geben Sie diesen Standort an, wenn Sie eine Anwendung starten.

Zugehörige Tasks

Anwendungen entwickeln

Zugehörige Verweise

„SOAP-Referenz” auf Seite 976
Die Referenzinformationen zu WebSphere MQ Transport for SOAP sind alphabetisch angeordnet.

„Referenzmaterial für IBM WebSphere MQ Bridge for HTTP” auf Seite 1242
Referenzthemen für IBM WebSphere MQ Bridge for HTTP in alphabetischer Reihenfolge

„IBM WebSphere MQ .NET-Klassen und -Schnittstellen” auf Seite 1278
Die .NET-Klassen und -Schnittstellen von IBM WebSphere MQ sind alphabetisch aufgelistet. Die Eigenschaften, Methoden und Konstruktoren werden beschrieben.

„IBM WebSphere MQ-C++-Klassen” auf Seite 1343
Die IBM WebSphere MQ-C++-Klassen binden die IBM WebSphere MQ Message Queue Interface (MQI) mit ein. Die einzelne C++-Headerdatei **imqi.hpp** deckt all diese Klassen ab.

WebSphere MQ-Klassen für Java Message Service

MQIEP-Struktur

Die MQIEP-Struktur enthält einen Eingangspunkt für jeden Funktionsaufruf, den Exits durchführen können.

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQIEP_STRUC_ID

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQIEP_VERSION_1
Strukturversionsnummer Version 1.

MQIEP_CURRENT_VERSION
Aktuelle Version der Struktur.

StrucLength
Typ: MQLONG

Größe der MQIEP-Struktur in Byte. Der Wert lautet wie folgt:

MQIEP_LENGTH_1

Markierungen
Typ: MQLONG

Stellt Informationen über die Funktionsadressen bereit. Ein Flag, um anzuzeigen, ob die Bibliothek über einen Thread verfügt und mit einem Flag verwendet werden kann, das anzeigt, ob es sich bei der Bibliothek um eine Client- oder Serverbibliothek handelt.

Der folgende Wert wird verwendet, um keine Bibliotheksinformationen anzugeben:

MQIEPF_NONE

Einer der folgenden Werte wird verwendet, um anzugeben, ob die gemeinsam genutzte Bibliothek über einen Thread verfügt oder nicht:

MQIEPF_NON_THREADED_LIBRARY
Eine gemeinsam genutzte Bibliothek ohne Thread

MQIEPF_THREADED_LIBRARY
Eine gemeinsam genutzte Bibliothek mit Thread

Einer der folgenden Werte wird verwendet, um anzugeben, ob es sich bei der gemeinsam genutzten Bibliothek um eine gemeinsam genutzte Client- oder Serverbibliothek handelt:

MQIEPF_CLIENT_LIBRARY
Eine gemeinsam genutzte Clientbibliothek

MQIEPF_LOCAL_LIBRARY
Eine gemeinsam genutzte Serverbibliothek

Reserved
Typ: MQPTR

MQBACK_Call
Typ: PMQ_BACK_CALL

Adresse des MQBACK-Aufrufs.

MQBEGIN_Call
Typ: PMQ_BEGIN_CALL

Adresse des MQBEGIN-Aufrufs.

MQBUFMH_Call
Typ: PMQ_BUFMH_CALL

Adresse des MQBUFMH-Aufrufs.

MQCB_Call
Typ: PMQ_CB_CALL

Adresse des MQCB-Aufrufs.

MQCLOSE_Call
Typ: PMQ_CLOSE_CALL

Adresse des MQCLOSE-Aufrufs.

MQCMIT_Call
Typ: PMQ_CMIT_CALL

Adresse des MQCMIT-Aufrufs.

MQCONN_Call
Typ: PMQ_CONN_CALL

Adresse des MQCONN-Aufrufs.

MQCONNX_Call
Typ: PMQ_CONNX_CALL

Adresse des MQCONNX-Aufrufs.

MQCRTMH_Call
Typ: PMQ_CRTMH_CALL

Adresse des MQCRTMH-Aufrufs.

MQCTL_Call
Typ: PMQ_CTL_CALL

Adresse des MQCTL-Aufrufs.

MQDISC_Call
Typ: PMQ_DISC_CALL

Adresse des MQDISC-Aufrufs.

MQDLTMH_Call
Typ: PMQ_DLTMH_CALL

Adresse des MQDLTMH-Aufrufs.

MQDLTMP_Call
Typ: PMQ_DLTMP_CALL

Adresse des MQDLTMP-Aufrufs.

MQGET_Call
Typ: PMQ_GET_CALL

Adresse des MQGET-Aufrufs.

MQINQ_Call
Typ: PMQ_INQ_CALL

Adresse des MQINQ-Aufrufs.

MQINQMP_Call
Typ: PMQ_INQMP_CALL

Adresse des MQINQMP-Aufrufs.

MQMHBUF_Call
Typ: PMQ_MHBUF_CALL

Adresse des MQMHBUF-Aufrufs.

MQOPEN_Call
Typ: PMQ_OPEN_CALL

Adresse des MQOPEN-Aufrufs.

MQPUT_Call
Typ: PMQ_PUT_CALL

Adresse des MQPUT-Aufrufs.

MQPUT1_Call
Typ: PMQ_PUT1_CALL

Adresse des MQPUT1-Aufrufs.

MQSET_Call
Typ: PMQ_SET_CALL

Adresse des MQSET-Aufrufs.

MQSETMP_Call
Typ: PMQ_SETMP_CALL

Adresse des MQSETMP-Aufrufs.

MQSTAT_Call
Typ: PMQ_STAT_CALL

Adresse des MQSTAT-Aufrufs.

MQSUB_Call
Typ: PMQ_SUB_CALL

Adresse des MQSUB-Aufrufs.

MQSUBRQ_Call
Typ: PMQ_SUBRQ_CALL

Adresse des MQSUBRQ-Aufrufs.

MQXCNVC_Call
Typ: PMQ_XCNVC_CALL

Adresse des MQXCNVC-Aufrufs.

MQXCLWLN_Call
Typ: PMQ_XCLWLN_CALL

Adresse des MQXCLWLN-Aufrufs.

MQXDX_Call
Typ: PMQ_XDX_CALL

Adresse des MQXDX-Aufrufs.

MQXEP_Call
Typ: PMQ_XEP_CALL

Adresse des MQXEP-Aufrufs.

MQZEP_Call
Typ: PMQ_ZEP_CALL

Adresse des MQZEP-Aufrufs.

C-Deklaration

```
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 */
 };
```

Datenkonvertierungsexit-Referenz

Bei z/OS müssen Sie Datenkonvertierungsexits in Assemblersprache schreiben. Bei anderen Plattformen wird empfohlen, die Programmiersprache C zu verwenden.

Für die Erstellung eines Datenkonvertierungsexitprogramms werden folgende Hilfen bereitgestellt:

- Eine Entwurfsquellendatei
- Ein Zeichenkonvertierungsaufruf
- Ein Dienstprogramm, das ein Fragment eines Codes erstellt, der die Datenkonvertierung auf Datentypstrukturen ausführt. Dieses Dienstprogramm akzeptiert ausschließlich Eingaben in C. Bei z/OS erstellt es Assemblercode.

Das Verfahren zum Schreiben des Programm finden Sie unter:

- Datenkonvertierungsexit für WebSphere MQ auf UNIX and Linux-Systemen schreiben
- Datenkonvertierungsexit für WebSphere MQ for Windows schreiben

Entwurfsquellendatei

Diese können beim Schreiben eines Datenkonvertierungsexitprogramms als Ausgangspunkt verwendet werden.

Die Dateien werden unter Tabelle 588 auf Seite 1040 bereitgestellt.

Tabelle 588. Entwurfsquellendateien	
Plattform	**Datei**
AIX	amqsvfc0.c
IBM i	QMQMSAMP/QCSRC(AMQSVFC4)
HP-UX	amqsvfc0.c
Linux	amqsvfc0.c
z/OS	CSQ4BAX8 („1” auf Seite 1040) CSQ4BAX9 („2” auf Seite 1040) CSQ4CAX9 („3” auf Seite 1040)
Solaris	amqsvfc0.c
Windows-Systeme	amqsvfc0.c
Anmerkungen: 1. Darstellung des MQXCVNC-Aufrufs. 2. Eine Oberfläche für Codefragmente, die vom Dienstprogramm für die Verwendung in allen Umgebungen erstellt wird, außer in CICS. 3. Eine Oberfläche für Codefragmente, die vom Dienstprogramm für die Verwendung in der CICS-Umgebung erstellt wird.	

Zeichenkonvertierungsaufruf

Verwenden Sie den MQXCNVC-Aufruf (Konvertierungszeichen) aus dem Datenkonvertierungsexitprogramm, um Zeichendaten aus Nachrichten von einem Zeichen in ein anderes umzuwandeln. Bei bestimmten Mehrbytezeichensätzen (z. B. UCS2-Zeichensätzen) müssen die entsprechenden Optionen verwendet werden.

Aus dem Exit können keine weiteren MQI-Aufrufe erfolgen; derartige Versuche schlagen mit dem Ursachencode MQRC_CALL_IN_PROGRESS fehl.

Im Abschnitt „MQXCNVC – Zeichen konvertieren” auf Seite 921 erhalten Sie weitere Informationen zum MQXCNVC-Aufruf und zu entsprechenden Optionen.

Dienstprogramm zum Einrichten eines Konvertierungsexitcodes

Dieser Abschnitt enthält Informationen über das Einrichten eines Konvertierungsexitcodes.

Die Befehle zum Erstellen von Konvertierungsexitcodes sind Folgende:

IBM i
CVTMQMDTA (Konvertiert WebSphere MQ-Datentyp)

Windows, UNIX and Linux-Systeme
`crtmqcvx` (erstellt WebSphere MQ-Konvertierungsexit)

Der Befehl für Ihre Plattform erzeugt ein Fragment eines Codes, der auf Datentypstrukturen Datenkonvertierung ausführt, den Sie in Ihrem Datenkonvertierungsprogramm verwenden können. Der Befehl nimmt eine Datei, die mindestens eine Strukturdefinition der Programmiersprache C enthält. .

Fehlernachrichten in Windows, UNIX and Linux-Systemen

Der Befehl `crtmqcvx` gibt Nachrichten im Bereich von AMQ7953 bis AMQ7970 zurück.

Diese Nachrichten werden unter Ursachencodes*WebSphere MQ-Nachrichten aufgelistet*.

Es gibt zwei wesentliche Fehlertypen:

- Schwerwiegende Fehler, z. B. Syntaxfehler, wenn die Verarbeitung nicht fortgesetzt werden kann.

  Auf dem Bildschirm wird eine Nachricht angezeigt, die die Zeilennummer des Fehlers in der Eingabedatei angibt. Die Ausgabedatei wurde möglicherweise nur teilweise erstellt.
- Bei anderen Fehlern wird eine Nachricht angezeigt, die besagt, dass ein Problem gefunden wurde, aber die Syntaxanalyse kann fortgesetzt werden.

  Die Ausgabedatei wurde erstellt und enthält Fehlerinformationen über die aufgetretenen Probleme. Diese Fehlerinformationen sind am Präfix `#error` erkennbar, damit der erzeugte Code von keinem Compiler ohne Eingriff zur Fehlerbehebung akzeptiert wird.

Gültige Syntax

Ihre Eingabedatei für das Dienstprogramm muss der Syntax der Programmiersprache C entsprechen.

Wenn Sie sich mit C nicht auskennen, sehen Sie sich das C Beispiel in diesem Abschnitt an.

Achten Sie darüber hinaus auf folgende Regeln:

- typedef wird nur vor dem Struct-Schlüsselwort erkannt.
- Bei Ihren Strukturdeklarationen ist eine Strukturkennung erforderlich.
- Sie können leere eckige Klammern [ ] verwenden, um eine variable Längenfeldgruppe oder Zeichenfolge am Ende der Nachricht anzuzeigen.
- Mehrdimensionale Feldgruppen und Feldgruppen von Zeichenfolgen werden nicht unterstützt.
- Es werden folgende zusätzliche Datentypen erkannt:
  - MQBOOL
  - MQBYTE
  - MQCHAR
  - MQFLOAT32
  - MQFLOAT64
  - MQSHORT
  - MQLONG
  - MQINT8
  - MQUINT8
  - MQINT16
  - MQUINT16
  - MQINT32
  - MQUINT32
  - MQINT64
  - MQUINT64

  MQCHAR-Felder sind gemäß Codepage konvertiert, allerdings werden MQBYTE, MQINT8 und MQUINT8 nicht berührt. Wenn die Codierung abweicht, werden MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 und MQBOOL dementsprechend konvertiert.
- Verwenden Sie *nicht* folgende Datentypen:
  - double

- Verweise
- Bitfelder

  Der Grund hierfür ist, dass das Dienstprogramm bei der Erstellung von Konvertierungsexitcode nicht die Möglichkeit bietet, diese Datentypen zu konvertieren. Wenn Sie dies umgehen möchten, schreiben Sie Ihre eigenen Routinen und rufen Sie sie über den Exit auf.

Weitere Punkte, die Sie beachten sollten:

- Verwenden Sie in der Eingabedatei keine Folgenummern.
- Wenn es Felder gibt, für die Sie Ihre eigene Konvertierungsroutine bereitstellen möchten, deklarieren Sie sie MQBYTE und ersetzen Sie anschließend die erstellten CMQXCFBA-Makros mit Ihrem eigenen Konvertierungscode.

Beispiel C

```
  struct TEST { MQLONG    SERIAL_NUMBER;
                MQCHAR    ID[5];
                MQINT16   VERSION;
                MQBYTE    CODE[4];
                MQLONG    DIMENSIONS[3];
                MQCHAR    NAME[24];
              } ;
```

Dies entspricht folgenden Deklarationen in anderen Programmiersprachen:

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

Nur unterstützt auf 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);
```

Veröffentlichungsexit - MQ_PUBLISH_EXIT

Durch den Aufruf MQ_PUBLISH_EXIT können die an Subskribenten zugestellten Nachrichten überprüft und geändert werden.

Verwendungszweck

Mithilfe des Veröffentlichungsexits können Sie die Nachrichten, die Subskribenten zugestellt werden, überprüfen und ändern:

- Inhalte einer Nachricht überprüfen, die für jeden Subskribenten veröffentlicht wird
- Inhalte einer Nachricht ändern, die für jeden Subskribenten veröffentlicht wird
- Warteschlange ändern, in die eine Nachricht eingereiht wird
- Übermittlung einer Nachricht an einen Subskribenten stoppen

Syntax

MQ_PUBLISH_EXIT(*ExitParms, PubContext, SubContext*)

Parameter

***ExitParms* (MQPSXP)- Input/Output**
ExitParms enthält Informationen zum Aufruf des Exits.

***PubContext* (MQPBC) - Input**
PubContext enthält Kontextinformationen zur Veröffentlichungskomponente der Veröffentlichung.

***SubContext* (MQSBC) - Input/Output**
SubContext enthält Kontextinformationen zum Subskribenten, der die Veröffentlichung empfängt.

MQPSXP - Datenstruktur des Veröffentlichungsexits

Die Struktur MQPSXP beschreibt die Informationen, die an den Veröffentlichungsexit übergeben und von diesem zurückgegeben werden.

Tabelle 589 auf Seite 1043 enthält eine Zusammenfassung der Felder in der Struktur:

Tabelle 589. Felder in MQPSXP

Feld	Beschreibung
StrucID	Struktur-ID
Version	Strukturversionsnummer
ExitId	Typ des aufgerufenen Exits
ExitReason	Ursache für Aufruf des Exits
ExitResponse	Antwort des Exits
ExitResponse2	Sekundäre Exit-Antwort
Feedback	Rückmeldungscode
ExitUserArea	Exit-Benutzerbereich
ExitData	Exitdaten
QMgrName	Name des lokalen Warteschlangenmanagers
Hconn	Verbindungskennung
MsgDescPtr	Adresse des Nachrichtendeskriptors (MQMD)
MsgHandle	Kennung für Nachrichteneigenschaften (MQHMSG)

Tabelle 589. Felder in MQPSXP (Forts.)	
Feld	**Beschreibung**
MsgInPtr	Adresse der Eingabenachricht
MsgInLength	Länge der Eingabenachricht
MsgOutPtr	Adresse der Ausgabenachricht
MsgOutLength	Länge der Ausgabenachricht
pEntryPoints	Adresse der MQIEP-Struktur

Felder

***StrucID* (MQCHAR4)**
StrucID steht für die Struktur-ID. Der Wert lautet wie folgt:

MQPSXP_STRUCID
MQPSXP_STRUCID ist die ID für die Parameterstruktur des Veröffentlichungsexits. Für die Programmiersprache C ist auch die Konstante MQPSXP_STRUC_ID_ARRAY definiert; sie hat denselben Wert wie MQPSXP_STRUC_ID, aber sie ist ein Array von Zeichen anstelle einer Zeichenfolge.

StrucID ist ein Eingabefeld für den Exit.

***Version* (MQLONG)**
Version ist die Strukturversionsnummer. Der Wert lautet wie folgt:

MQPSXP_VERSION_1
MQPSXP_VERSION_1 ist Version 1 der Parameterstruktur des Veröffentlichungsexits. Die Konstante MQPSXP_CURRENT_VERSION ist ebenfalls mit demselben Wert definiert.

Version ist ein Eingabefeld für den Exit.

***ExitId* (MQLONG)**
ExitId ist der Typ des aufgerufenen Exits. Der Wert lautet wie folgt:

MQXT_PUBLISH_EXIT
Veröffentlichungsexit.

ExitId ist ein Eingabefeld für den Exit.

***ExitReason* (MQLONG)**
ExitReason ist die Ursache für den Aufruf des Exits. Folgende Werte sind möglich:

MQXR_INIT
Der Exit für diese Verbindung wird zur Initialisierung aufgerufen. Der Exit fordert gegebenenfalls die benötigten Ressourcen, z. B. Hauptspeicher, an und initialisiert sie.

MQXR_TERM
Der Exit für diese Verbindung wird aufgerufen, weil der Exit gestoppt werden soll. Der Exit muss alle Ressourcen freigeben, die er seit seiner Initialisierung angefordert hat, z. B. Hauptspeicher.

MQXR_PUBLICATION
Der Exit wird vom Warteschlangenmanager aufgerufen, bevor er eine Veröffentlichung in eine Nachrichtenwarteschlange eines Subskribenten einreiht. Der Exit kann die Nachricht ändern, die Nachricht nicht in die Warteschlange einreihen oder die Veröffentlichung stoppen.

ExitReason ist ein Eingabefeld für den Exit.

***ExitResponse* (MQLONG)**
Geben Sie *ExitResponse* im Exit an, um die Vorgehensweise für die weitere Verarbeitung festzulegen. *ExitResponse* ist einer der folgenden Werte:

MQXCC_OK
Geben Sie MQXCC_OK an, um die Verarbeitung normal fortzusetzen. Setzen Sie MQXCC_OK als Antwort auf alle Werte von ExitReason.

Wenn ExitReason den Wert MQXR_PUBLICATION hat, geben die Felder *DestinationQName* und *DestinationQMgrName* der MQSBC -Struktur das Ziel an, an das die Nachricht gesendet wird.

MQXCC_FAILED
Geben Sie MQXCC_FAILED an, um den Veröffentlichungsvorgang zu stoppen. Der Beendigungscode MQCC_FAILED und der Ursachencode 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR werden für die Rückgabe vom Exit festgelegt.

MQXCC_SUPPRESS_FUNCTION
Geben Sie MQXCC_SUPPRESS_FUNCTION an, um die normale Verarbeitung der Nachricht zu stoppen. Legen Sie fürMQXCC_SUPPRESS_FUNCTION nur dann einen Wert fest, wenn *ExitReason* den Wert MQXR_PUBLICATION hat.

Die Nachricht wird weiterhin vom WS-Manager gemäß der Option MQRO_DISCARD_MSG im Feld *Report* im Nachrichtendeskriptor der Nachricht verarbeitet.

- Wenn die Option MQRO_DISCARD_MSG angegeben ist, wird die Nachricht nicht an den Subskribenten übermittelt.
- Wenn die Option MQRO_DISCARD_MSG nicht angegeben ist, wird die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht. Wenn keine Warteschlange für nicht zustellbare Nachrichten vorhanden ist oder die Nachricht nicht erfolgreich in die Warteschlange für nicht zustellbare Nachrichten eingereiht werden kann, wird die Veröffentlichung nicht an den Subskribenten übermittelt. Die Zustellung der Veröffentlichung an andere Subskribenten hängt von den Werten der Themenobjektattribute PMSGDLV und NPMSGDLV ab. Eine Erläuterung dieser Attribute finden Sie in den Parameterbeschreibungen für den Befehl DEFINE TOPIC.

ExitResponse ist ein Ausgabefeld für den Exit.

***ExitResponse2* (MQLONG)**
ExitResponse2 ist für eine spätere Verwendung reserviert.

***Feedback* (MQLONG)**
Feedback ist der Rückkopplungscode, der verwendet wird, wenn der Exit den Wert MQXCC_SUPPRESS_FUNCTION in *ExitResponse* zurückgibt.

Bei der Eingabe für den Exit hat *Feedback* immer den Wert MQFB_NONE. Wenn der Exit MQXCC_SUPPRESS_FUNCTIONzurückgibt, setzen Sie *Feedback* auf den Wert, der für die Nachricht verwendet werden soll, wenn der WS-Manager sie in die Warteschlange für nicht zustellbare Nachrichten einreiht. Wenn *Feedback* bei der Rückkehr vom Exit den ursprünglichen Wert MQFB_NONEhat, setzt der Warteschlangenmanager *Feedback* auf MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback ist ein Ein-/Ausgabefeld für den Exit.

***ExitUserArea* (MQBYTE16)**
ExitUserArea ist ein Feld, das zur Verwendung durch den Exit zur Verfügung steht. Jede Verbindung hat eine separate *ExitUserArea*. Die Länge von *ExitUserArea* wird durch MQ_EXIT_USER_AREA_LENGTH angegeben.

Das Feld *ExitReason* enthält beim ersten Aufruf des Exits den Wert MQXR_INIT. *ExitUserArea* wird beim ersten Aufruf des Exits für eine Verbindung mit MQXUA_NONE initialisiert. Nachfolgende Änderungen von *ExitUserArea* bleiben über Aufrufe des Exits hinweg bestehen.

ExitUserArea ist ein Ein-/Ausgabefeld für den Exit.

***ExitData* (MQCHAR32)**
ExitData enthält Exitdaten, die durch den Parameter *PublishExitData* in der Zeilengruppe in der Initialisierungsdatei des Warteschlangenmanagers festgelegt sind. Die Daten werden bis zur vollen Länge des Feldes mit Leerzeichen aufgefüllt. Wenn in der Initialisierungsdatei keine Exitdaten festgelegt sind, enthält *ExitData* nur Leerzeichen. Die Länge von *ExitData* wird durch MQ_EXIT_DATA_LENGTHangegeben.

ExitData ist ein Eingabefeld für den Exit.

***QMgrName* (MQCHAR48)**
QMgrName ist der Name des lokalen Warteschlangenmanagers. Der Name ist auf der gesamten Länge des Felds mit Leerzeichen aufgefüllt worden. Die Länge dieses Feldes ist durch MQ_Q_MGR_NAME_LENGTH angegeben.

QMgrName ist ein Eingabefeld für den Exit.

***Hconn* (MQHCONN)**
Hconn ist die Kennung für die Verbindung zum Warteschlangenmanager. Verwenden Sie nur *Hconn* als Parameter für die Funktionsaufrufe der Nachrichteneigenschaft MQSETMP, MQINQMMPoder MQDLTMP , um mit Nachrichteneigenschaften zu arbeiten.

Hconn ist ein Eingabefeld für den Exit.

***MsgDescPtr* (PMQMD)**
MsgDescPtr ist die Adresse des Nachrichtendeskriptors (MQMD) der zu verarbeitenden Nachricht und eine Kopie des Nachrichtendeskriptors MQMD, der vom MQPUT-Aufruf zurückgegeben wird. Der Exit kann den Inhalt des Nachrichtendeskriptors ändern. Bei Änderungen des Inhalts des Nachrichtendeskriptors ist Vorsicht geboten. Insbesondere wenn das Feld *SubType* der Struktur MQSBC den Wert MQSUBTYPE_PROXYhat, darf das Feld *CorrelId* im Nachrichtendeskriptor nicht geändert werden.

Wenn *ExitReason* auf MQXR_INIT oder MQXR_TERM gesetzt ist, wird kein Nachrichtendeskriptor an den Exit übergeben. In diesen Fällen ist *MsgDescPtr* der Nullzeiger.

MsgDescPtr ist ein Eingabefeld für den Exit.

***MsgHandle* (MQHMSG)**
MsgHandle ist die Kennung für Nachrichteneigenschaften. Verwenden Sie *MsgHandle* nur mit den Nachrichteneigenschaftenfunktionsaufrufen MQSETMP, MQINQMMP und MQDLTMP, um Nachrichteneigenschaften zu bearbeiten.

MsgHandle ist ein Eingabefeld für den Exit.

***MsgInPtr* (PMQVOID)**
MsgInPtr ist die Adresse der Eingabenachrichtendaten. Der Inhalt des von *MsgInPtr* adressierten Puffers kann vom Exit geändert werden (siehe *MsgOutPtr*).

MsgInPtr ist ein Eingabefeld für den Exit.

***MsgInLength* (MQLONG)**
MsgInLength ist die Länge der an den Exit übergebenen Nachrichtendaten in Byte. Die Adresse der Daten wird durch *MsgInPtr* angegeben.

MsgInLength ist ein Eingabefeld für den Exit.

***MsgOutPtr* (PMQVOID)**
MsgOutPtr ist die Adresse eines Puffers mit den Nachrichtendaten, die vom Exit zurückgegeben werden. Bei der Eingabe für den Exit ist *MsgOutPtr* auf 0 gesetzt. Wenn der Wert bei Rückgabe des Exits immer noch null ist, sendet der Warteschlangenmanager die Nachricht, die in *MsgInPtr* angegeben wurde, mit der in *MsgInLength* angegebenen Länge.

Wenn der Exit die Nachrichtendaten ändert, verwenden Sie eine der folgenden Vorgehensweisen:

- Wenn sich die Länge der Daten nicht ändert, können die Daten in dem durch *MsgInPtr* angegebenen Puffer geändert werden. Ändern Sie in diesem Fall nicht *MsgOutPtr* und *MsgOutLength*.
- Wenn die geänderten Daten kürzer sind als die ursprünglichen Daten, können die Daten in dem von *MsgInPtr* adressierten Puffer geändert werden. In diesem Fall muss *MsgOutPtr* auf die Adresse des Eingabenachrichtenpuffers und *MsgOutLength* auf die neue Länge der Nachrichtendaten gesetzt werden.
- Wenn die geänderten Daten tatsächlich oder möglicherweise länger als die ursprünglichen Daten sind, muss der Exit einen neuen Nachrichtenpuffer anfordern. Kopieren Sie die geänderten Daten in den neuen Puffer. Setzen Sie *MsgOutPtr* auf die Adresse des neuen Puffers und *MsgOutLength* auf die Länge der neuen Nachrichtendaten. Der Exit ist dafür verantwortlich, den von *MsgOutPtr* adressierten Puffer freizugeben, wenn der Exit das nächste Mal aufgerufen wird.

Anmerkung: *MsgOutPtr* ist bei der Eingabe für den Exit immer der Nullzeiger und nicht die Adresse eines zuvor angeforderten Nachrichtenpuffers. Um den zuvor angeforderten Puffer freigeben zu können, muss der Exit dessen Adresse und Länge speichern. Speichern Sie die Informationen entweder in *ExitUserArea* oder in einem Steuerblock, dessen Adresse in *ExitUserArea* gespeichert ist.

MsgOutPtr ist ein Ein-/Ausgabefeld für den Exit.

***MsgOutLength* (MQLONG)**
MsgOutLength ist die Länge der vom Exit zurückgegebenen Nachrichtendaten in Byte. Bei der Eingabe für den Exit ist dieses Feld immer auf 0 gesetzt. Bei der Rückgabe vom Exit wird das Feld ignoriert, wenn *MsgOutPtr* 0 ist. Informationen zum Ändern der Nachrichtendaten finden Sie unter *MsgOutPtr* .

MsgOutLength ist ein Ein-/Ausgabefeld für den Exit.

***pEntryPoints* (PMQIEP)**
pEntryPoints ist die Adresse einer MQIEP-Struktur, über die MQI- und DCI-Aufrufe ausgegeben werden können.

Deklaration in Programmiersprache 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;
```

Kontextdatenstruktur der Veröffentlichung - MQPBC

Die MQPBC-Struktur enthält Kontextinformationen zur Veröffentlichungskomponente der Veröffentlichung, die an den Veröffentlichungsexit übergeben werden.

Tabelle 590 auf Seite 1047 enthält eine Zusammenfassung der Felder in der Struktur:

Tabelle 590. Felder in MQPBC

Feld	Beschreibung
StrucID	Struktur-ID
Version	Strukturversionsnummer
PubTopicString	Themenzeichenfolge veröffentlichen
MsgDescPtr	Adresse des Nachrichtendeskriptors (MQMD)

Felder

***StrucID* (MQCHAR4)**
StrucID steht für die Struktur-ID. Der Wert lautet wie folgt:

MQPBC_STRUCID
MQPBC_STRUCID steht für die Kontextstruktur-ID der Veröffentlichung. Für die Programmiersprache C ist auch die Konstante MQPBC_STRUC_ID_ARRAY definiert; diese Konstante hat den gleichen Wert wie MQPBC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

StrucID ist ein Eingabefeld für den Exit.

***Version* (MQLONG)**
Version ist die Strukturversionsnummer. Der Wert lautet wie folgt:

MQPBC_VERSION_1
MQPBC_VERSION_1 steht für Version 1 der Parameterstruktur des Veröffentlichungsexits.

MQPBC_VERSION_2
MQPBC_VERSION_2 steht für Version 2 der Parameterstruktur des Veröffentlichungsexits. Die Konstante MQPBC_CURRENT_VERSION wird mit dem gleichen Wert definiert.

Version ist ein Eingabefeld für den Exit.

***PubTopicString* (MQCHARV)**
PubTopicString steht für die Themenzeichenfolge, zu der veröffentlicht wird.

PubTopicString ist ein Eingabefeld für den Exit.

***MsgDescPtr* (PMQMD)**
MsgDescPtr ist die Adresse einer Kopie des Nachrichtendeskriptors (MQMD) für die Nachricht, die gerade verarbeitet wird.

MsgDescPtr ist ein Eingabefeld für den Exit.

Deklaration in Programmiersprache 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;
```

Kontextdatenstruktur der Subskription - MQSBC

Die MQSBC-Struktur enthält Kontextinformationen zu dem Subskribenten der Veröffentlichung, die an den Veröffentlichungsexit übergeben werden.

Tabelle 591 auf Seite 1048 enthält eine Zusammenfassung der Felder in der Struktur:

Tabelle 591. Felder in MQSBC

Feld	Beschreibung
StrucID	Struktur-ID
Version	Strukturversionsnummer
DestinationQMgrName	Name des Ziel-Warteschlangenmanagers
DestinationQName	Name der Zielwarteschlange
SubType	Subskriptionstyp
SubOptions	Subskriptionsoptionen
ObjectName	Objektname
ObjectString	Objektzeichenfolge
SubTopicString	Themenzeichenfolge der Subskription

Tabelle 591. Felder in MQSBC (Forts.)	
Feld	**Beschreibung**
SubName	Name der Subskription
SubId	Subskriptionskennung
SelectionString	Adresse der Auswahlzeichenfolge
SubLevel	Subskriptionsebene
PSProperties	Publish/Subscribe-Eigenschaften

Felder

***StrucID* (MQCHAR4)**
Struktur-ID. Der Wert lautet wie folgt:

MQSBC_STRUCID
MQSBC_STRUCID ist die ID der Parameterstruktur des Veröffentlichungsexits. Für die Programmiersprache C ist auch die Konstante MQSBC_STRUC_ID_ARRAY definiert; MQSBC_STRUC_ID_ARRAY hat den gleichen Wert wie MQSBC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

StrucID ist ein Eingabefeld für den Exit.

***Version* (MQLONG)**
Strukturversionsnummer. Der Wert lautet wie folgt:

MQSBC_VERSION_1
Parameterstruktur des Veröffentlichungsexits Version 1. Die Konstante MQSBC_CURRENT_VERSION wird mit dem gleichen Wert definiert.

Version ist ein Eingabefeld für den Exit.

***DestinationQMgrName* (MQCHAR48)**
DestinationQMgrName ist der Name des Warteschlangenmanagers, an den die Nachricht gesendet wird. Der Name ist auf der gesamten Länge des Felds mit Leerzeichen aufgefüllt worden. Der Exit kann den Namen ändern. Die Länge dieses Feldes ist durch MQ_Q_MGR_NAME_LENGTH angegeben.

DestinationQMgrName ist ein Ein-/Ausgabefeld für den Exit; siehe hierzu die Anmerkung.

***DestinationQName* (MQCHAR48)**
DestinationQName ist der Name der Warteschlange, an die die Nachricht gesendet wird. Der Name ist auf der gesamten Länge des Felds mit Leerzeichen aufgefüllt worden. Der Exit kann den Namen ändern. Die Länge dieses Feldes ist durch MQ_Q_NAME_LENGTH angegeben.

DestinationQName ist ein Ein-/Ausgabefeld für den Exit; siehe hierzu die Anmerkung.

***SubType* (MQLONG)**
SubType gibt an, wie die Subskription erstellt wurde. Gültige Werte sind MQSUBTYPE_API, MQSUBTYPE_ADMIN und MQSUBTYPE_PROXY; weitere Informationen finden Sie unter Inquire Subscription Status (Antwort).

SubType ist ein Eingabefeld für den Exit.

***SubOptions* (MQLONG)**
SubOptions sind die Subskriptionsoptionen. Im Abschnitt „Optionen (MQLONG)" auf Seite 556 finden Sie eine Beschreibung der Werte, die für dieses Feld angegeben werden können.

SubOptions ist ein Eingabefeld für den Exit.

***ObjectName* (MQCHAR48)**
ObjectName ist der Name des Themenobjekts entsprechend der Definition auf dem lokalen Warteschlangenmanager. Die Länge dieses Feldes ist durch MQ_TOPIC_NAME_LENGTH angegeben. Der Objektname ist der Name des Verwaltungsthemenobjekts, das der Warteschlangenmanager der The-

menzeichenfolge zugeordnet hat. Auch wenn der Subskribent als Teil der Subskription ein Themenobjekt angegeben hat, ist *ObjectName* möglicherweise ein anderes Themenobjekt. Die Zuordnung eines Themenobjekts zu einer Subskription ist abhängig von der vollständigen Auflösung des Attributs *SubTopicString*.

ObjectName ist ein Eingabefeld für den Exit.

***ObjectString* (MQCHARV)**
ObjectString ist die vollständige Themenzeichenfolge der Veröffentlichung, die subskribiert wurde. Alle Platzhalterzeichen der ursprünglichen Subskriptionszeichenfolge wurden aufgelöst. Dieses Feld unterscheidet sich von dem in Abschnitt „ObjectString (MQCHARV)" auf Seite 555 beschriebenen *ObjectString*-Feld der MQSD-Subskription, das Platzhalterzeichen enthalten kann, abgesehen von Objektnamen, die vom Subskribenten angegeben werden.

ObjectString ist ein Eingabefeld für den Exit.

***SubTopicString* (MQCHARV)**
SubTopicString ist die vollständige Themenzeichenfolge, wie sie vom Subskribenten bereitgestellt wurde. *SubTopicString* ist die Kombination aus der in einem Themenobjekt definierten Themenzeichenfolge und einer Themenzeichenfolge. Ein Subskribent muss entweder ein Themenobjekt oder eine Themenzeichenfolge oder beides angeben. Wenn der Subskribent eine Themenzeichenfolge angibt, kann sie Platzhalterzeichen enthalten.

SubTopicString ist ein Eingabefeld für den Exit.

***SubName* (MQCHARV)**
SubName ist der Subskriptionsname, der vom Subskribenten angegeben wird, es sei denn, es ist ein generierter Name.

SubName ist ein Eingabefeld für den Exit.

***SubId* (MQBYTE 24)**
SubId steht für die eindeutige interne ID der Subskription.

SubId ist ein Eingabefeld für den Exit.

***SelectionString* (MQCHARV)**
SelectionString sind die Auswahlkriterien, die beim Subskribieren von Nachrichten aus einem Thema verwendet werden; sieheSelektoren .

SelectionString ist ein Eingabefeld für den Exit.

***SubLevel* (MQLONG)**
SubLevel ist die der Subskription zugeordnete Abfangebene; weitere Informationen hierzu finden Sie im Abschnitt „SubLevel (MQLONG)" auf Seite 568.

SubLevel ist ein Eingabefeld für den Exit.

***PSProperties* (MQLONG)**
PSProperties steht für die Publish/Subscribe-Eigenschaften. Diese Eigenschaften geben an, auf welche Art und Weise mit Publish/Subscribe zusammenhängende Nachrichteneigenschaften zu Nachrichten, die an diese Subskription gesendet werden, hinzugefügt werden. Mögliche Werte sind MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Eine Beschreibung dieser Werte finden Sie unter Optionale Parameter (Change, Copy und Create Subscription) .

PSProperties ist ein Eingabefeld für den Exit.

Anmerkung: Eine Berechtigungsprüfung wird nur für die ursprünglichen Werte von *DestinationQMgrName* und *DestinationQName* vorgenommen, bevor diese an den Veröffentlichungsexit übermittelt werden. Wenn der Exit entweder durch eine Änderung an *DestinationQMgrName* oder an *DestinationQName* eine neue Zielwarteschlange angibt, wird keine erneute Berechtigungsprüfung durchgeführt.

Deklaration in Programmiersprache 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;
```

Kanalexitaufrufe und Datenstrukturen

Diese Themensammlung enthält Referenzinformationen zu besonderen WebSphere MQ-Aufrufen und Datenstrukturen, die beim Schreiben von Kanalexitprogrammen verwendet werden können.

Bei diesen Informationen handelt es sich um produktabhängige Informationen zur Programmierschnittstelle. WebSphere MQ-Benutzerexits können in folgenden Programmiersprachen geschrieben werden:

Plattform	Programmiersprachen
WebSphere MQ for z/OS	Assembler und C (diese müssen der im Handbuch *z/OS C/C++ Programming Guide* beschriebenen C-Systemprogrammierumgebung für Systemexits entsprechen.)
WebSphere MQ für IBM i	ILE C, ILE COBOL und ILE RPG
Alle anderen WebSphere MQ-Plattformen	C

Sie können auch Benutzerexits in Java für die ausschließliche Verwendung mit Java-und JMS-Anwendungen schreiben. Weitere Informationen zum Erstellen und Verwenden von Kanalexits mit den WebSphere MQ -Klassen für Java finden Sie unter Kanalexits in WebSphere MQ -Klassen für Java und für WebSphere MQ -Klassen für JMS unter Kanalexits mit WebSphere MQ -Klassen für JMS verwenden.

WebSphere MQ-Benutzerexits können nicht in TAL oder Visual Basic geschrieben werden. In Visual Basic wird jedoch eine Deklaration für die MQCD-Struktur bereitgestellt, die im MQCONNX-Aufruf von einem MQI-Clientprogramm von WebSphere MQ verwendet werden kann.

In einigen der folgenden Beschreibungen handelt es sich bei den Parametern um Arrays oder Zeichenfolgen mit einer nicht festgelegten Größe. Für diese Parameter wird für die Darstellung einer numerischen Konstante der Buchstabe "n" (in Kleinschreibung) verwendet. Wenn die Deklaration für diesen Parameter codiert ist, muss "n" durch den erforderlichen numerischen Wert ersetzt werden. Weitere Informationen zu den in diesen Beschreibungen verwendeten Konventionen finden Sie im Abschnitt „Elementardatentypen” auf Seite 218.

Datendefinitionsdateien

WebSphere MQ stellt für alle unterstützten Programmiersprachen Datendefinitionsdateien bereit. Ausführliche Informationen zu diesen Dateien finden Sie im Abschnitt Kopier-, Header-, Include- und Moduldateien.

MQ_CHANNEL_EXIT - Kanalexit

Der Aufruf MQ_CHANNEL_EXIT beschreibt die Parameter, die an die einzelnen, vom Nachrichtenkanalagenten aufgerufenen Kanalexits übermittelt werden.

Der Warteschlangenmanager stellt keinen Eingangspunkt namens MQ_CHANNEL_EXIT bereit. Der Name MQ_CHANNEL_EXIT hat keine besondere Bedeutung, da die Namen der Kanalexits in der Kanaldefinition MQCD vorgegeben werden.

Es gibt fünf Arten von Kanalexits:

- Kanalsicherheitsexit
- Kanalnachrichtenexit
- Kanalsenderexit
- Kanalempfangsexit
- Kanalexit für Nachrichtenwiederholungen

Die Exits verwenden ähnliche Parameter und die hier enthaltene Beschreibung gilt, sofern nicht anderweitig erwähnt, für alle Exits.

Syntax

MQ_CHANNEL_EXIT *(ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)*

Parameter

Der Aufruf MQ_CHANNEL_EXIT hat folgende Parameter:

ChannelExitParms (MQCXP) - Ein-/Ausgabe

Parameterblock des Kanalexits.

Diese Struktur enthält zusätzliche Informationen zum Aufrufen des Exits.Der Exit legt Informationen in dieser Struktur fest, um den Fortschritt des Nachrichtenkanalagenten anzuzeigen.

ChannelDefinition (MQCD) - Ein-/Ausgabe

Kanaldefinition.

Diese Struktur enthält die vom Administrator zur Steuerung des Kanalverhaltens festgelegten Parameter.

DataLength (MQLONG) - Ein-/Ausgabe

Länge der Daten.

Die Daten hängen von der Art des Exits ab:

- Wenn ein Kanalsicherheitsexit aufgerufen wird, enthält dieser Parameter die Länge jeder Sicherheitsnachricht im Feld *AgentBuffer*, wenn *ExitReason* auf MQXR_SEC_MSG gesetzt ist. Das Feld hat den Wert Null, wenn keine Nachricht vorliegt. Der Exit muss dieses Feld auf die Länge jeder Sicherheitsnachricht setzen, die an seinen Partner gesendet wird, wenn er *ExitResponse* auf MQXCC_SEND_SEC_MSG oder MQXCC_SEND_AND_REQUEST_SEC_MSG setzt. Die Nachrichtendaten sind entweder in *AgentBuffer* oder in *ExitBufferAddr* enthalten.

  Der Inhalt der Sicherheitsnachrichten unterliegt der alleinigen Zuständigkeit der Sicherheitsexits.
- Wenn ein Kanalnachrichtenexit aufgerufen wird, enthält dieser Parameter die Länge jeder Sicherheitsnachricht (einschließlich Header der Übertragungswarteschlange). Abhängig davon, mit welchem Feld fortgefahren wird, muss der Exit dieses Feld in *AgentBuffer* oder in *ExitBufferAddr* die Länge der Nachricht setzen. Der Wert muss größer-gleich der Länge des Headers der Übertragungswarteschlange sein (MQXQH).
- Wenn ein Kanalsender- bzw. Kanalempfangsexit aufgerufen wird, enthält dieser Parameter die Länge der Übertragung. Abhängig davon, mit welchem Feld fortgefahren wird, muss der Exit dieses Feld entweder in *AgentBuffer* oder in *ExitBufferAddr* auf die Länge der Übertragung setzen.

Wenn ein Sicherheitsexit eine Nachricht sendet und am anderen Kanalende kein Sicherheitsexit vorhanden ist bzw. das andere Kanalende *ExitResponse* auf MQXCC_OK setzt, wird der initialisierende Exit neu mit MQXR_SEC_MSG und einer Nullantwort (*DataLength*=0) aufgerufen.

AgentBufferLength (MQLONG) - Eingabe

Länge des Agentenpuffers.

Dieser Parameter kann beim Aufruf größer sein als *DataLength*.

Bei Kanalnachrichten-, Sende- und Empfangsexits kann der Exit beim Aufrufen jeden unbelegten Platz verwenden, um die Daten dort einzublenden. In diesem Fall muss er den Parameter *DataLength* entsprechend setzen.

In der Programmiersprache C wird dieser Parameter nach Adresse übergeben.

AgentBuffer (MQBYTE×AgentBufferLength) - Ein-/Ausgabe

Agentenpuffer.

Der Inhalt dieses Parameters hängt von der Art des Exits ab:

- Wenn ein Kanalsicherheitsexit aufgerufen wird, enthält dieser Parameter eine Sicherheitsnachricht, wenn *ExitReason* auf MQXR_SEC_MSG gesetzt ist. Zum Zurücksenden einer Nachricht kann der Exit entweder diesen oder seinen eigenen Puffer (*ExitBufferAddr*) verwenden.
- Wenn ein Kanalnachrichtenexit aufgerufen wird, enthält dieser Parameter Folgendes:
  - Header der Übertragungswarteschlange (MQXQH) mit dem Nachrichtendeskriptor (der wiederum die Kontextinformationen für die Nachricht enthält), unmittelbar gefolgt von
  - Nachrichtendaten

  Wenn die Nachricht fortgesetzt werden soll, kann der Exit eine der folgenden Aktionen ausführen:

  - Den Inhalt des Puffers unverändert lassen
  - Den verwendeten Inhalt ändern (Rückgabe der neuen Länge der Daten in *DataLength*. Dieser Wert muss größer sein als *AgentBufferLength*).
  - Den Inhalt in die *ExitBufferAddr* kopieren und dabei erforderliche Änderungen vornehmen

  Sämtliche Änderungen, die der Exit am Header der Übertragungswarteschlange vornimmt, werden nicht überprüft. Fehlerhafte Änderungen könnten jedoch dazu führen, dass die Nachricht nicht beim Empfänger eingereiht werden kann.
- Wenn ein Kanalsender- oder -empfangsexit aufgerufen wird, enthält dieser Parameter die Übertragungsdaten. Der Exit kann eine der folgenden Aktionen ausführen:
  - Den Inhalt des Puffers unverändert lassen
  - Den verwendeten Inhalt ändern (Rückgabe der neuen Länge der Daten in *DataLength*. Dieser Wert muss größer sein als *AgentBufferLength*).
  - Den Inhalt in die *ExitBufferAddr* kopieren und dabei erforderliche Änderungen vornehmen

  Die ersten 8 Byte der Daten dürfen vom Exit nicht geändert werden.

ExitBufferLength (MQLONG) - Ein-/Ausgabe

Länge des Exitpuffers.

Beim ersten Aufruf des Exits wird dieser Parameter auf null gesetzt. Danach wird dem Exit bei folgenden Aufrufen jeweils der Wert angezeigt, den der Exit zurückgegeben hat. Der Wert wird nicht vom Nachrichtenkanalagenten verwendet.

Anmerkung: Dieser Parameter darf nicht von Exits verwendet werden, die in Programmiersprachen geschrieben sind, welche den Datentyp "Pointer" nicht unterstützen.

ExitBufferAddr (MQPTR) - Ein-/Ausgabe

Adresse des Exitpuffers.

Dieser Parameter verweist auf die Pufferadresse eines vom Exit verwalteten Speichers, wo wahlweise Nachrichten- oder Übertragungsdaten (je nach Art des Exits) an den Agenten zurückgegeben werden können, wenn der Puffer des Agenten nicht groß genug ist/sein könnte oder wenn dies für den Exit einfacher zu handhaben ist.

Beim ersten Aufruf des Exits lautet die an den Exit übergebene Adresse Null. Danach wird dem Exit bei folgenden Aufrufen jeweils die Adresse angezeigt, die der Exit zurückgegeben hat.

Anmerkung: Dieser Parameter darf nicht von Exits verwendet werden, die in Programmiersprachen geschrieben sind, welche den Datentyp "Pointer" nicht unterstützen.

C-Aufruf

```
exitname (&ChannelExitParms, &ChannelDefinition,
         &DataLength, &AgentBufferLength, AgentBuffer,
         &ExitBufferLength, &ExitBufferAddr);
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
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 */
```

Aufruf in COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                           DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                           EXITBUFFERLENGTH, EXITBUFFERADDR.
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
**   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.
```

RPG-Aufrufe (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD : DATLEN :
     C                                       ABUFL : ABUF : EBUFL :
     C                                       EBUF)
```

Die Prototypdefinition für den Aufruf ist:

```
     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                            *
```

System/390 -Assembleraufruf

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X
               AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH,         X
               EXITBUFFERADDR)
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
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
```

Hinweise zur Verwendung

1. Die vom Kanalexit ausgeführte Funktion wird vom Exit-Provider definiert. Der Exit muss jedoch den hier und im zugehörigen Steuerblock (MQCXP) definierten Regeln entsprechen.
2. Der an den Kanalexit übergebene Parameter *ChannelDefinition* kann verschiedene Versionen haben. Weitere Informationen finden Sie in der MQCD-Struktur im Feld *Version*.
3. Wenn der Kanalexit eine MQCD-Struktur empfängt, in der der Wert im Feld *Version* größer ist als MQCD_VERSION_1, muss der Exit statt dem Feld *ShortConnectionName* das Feld *ConnectionName* in MQCD verwenden.
4. Generell dürfen Kanalexits die Länge der Nachrichtendaten ändern. Dies kann erforderlich sein, wenn der Exit der Nachricht Daten hinzufügt, Daten aus der Nachricht entfernt oder die Nachricht verschlüsselt. Es gelten jedoch besondere Einschränkungen, wenn es sich bei der Nachricht um ein Segment handelt, das nur Teile der logischen Nachricht enthält. Insbesondere darf es als Ergebnis der Aktionen ergänzender Sender- und Empfangsexits keine Nettoänderung in der Länge der Nachricht geben.

   Ein Senderexit darf beispielsweise die Nachricht durch Komprimierung verkürzen, doch der ergänzende Empfangsexit muss die Originallänge der Nachricht wiederherstellen, indem er die Nachricht dekomprimiert, damit es zu keiner Nettoänderung der Nachrichtenlänge kommt.

   Es kommt zu dieser Einschränkung, da durch Ändern der Länge eines Segments die Offsets späterer Segmenten in der Nachricht falsch sind. Dadurch kann der Warteschlangenmanager nicht erkennen, dass die Segmente eine vollständige logische Nachricht gebildet haben.

MQ_CHANNEL_AUTO_DEF_EXIT - Exit für die automatische Kanaldefinition

Der Aufruf MQ_CHANNEL_AUTO_DEF_EXIT beschreibt die Parameter, die an den vom Nachrichtenkanalagenten aufgerufenen Exit für die automatische Kanaldefinition übermittelt werden.

Der Warteschlangenmanager stellt keinen Eingangspunkt namens MQ_CHANNEL_AUTO_DEF_EXIT bereit. Der Name MQ_CHANNEL_AUTO_DEF_EXIT hat keine besondere Bedeutung, da die Namen der Exits für die automatische Kanaldefinition im Warteschlangenmanager vorgegeben sind.

Syntax

MQ_CHANNEL_AUTO_DEF_EXIT *(ChannelExitParms, ChannelDefinition)*

Parameter

Der Aufruf MQ_CHANNEL_AUTO_DEF_EXIT hat folgende Parameter:

ChannelExitParms (MQCXP) - Ein-/Ausgabe

Parameterblock des Kanalexits.

Diese Struktur enthält zusätzliche Informationen zum Aufrufen des Exits.Der Exit legt Informationen in dieser Struktur fest, um den Fortschritt des Nachrichtenkanalagenten anzuzeigen.

ChannelDefinition (MQCD) - Ein-/Ausgabe

Kanaldefinition.

Diese Struktur enthält die vom Administrator zur Steuerung des Verhaltens von automatisch erstellten Kanälen festgelegten Parameter.Der Exit legt Informationen in dieser Struktur fest, um das vom Administrator festgelegte Standardverhalten zu ändern.

Die folgenden MQCD-Felder dürfen vom Exit nicht geändert werden:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Wenn andere Felder geändert werden, muss der vom Exit gesetzte Wert gültig sein. Ist der Wert nicht gültig, wird eine Fehlernachricht in die in der Konsole angezeigte Fehlerprotokolldatei geschrieben oder auf der Konsole angezeigt (je nach verwendeter Umgebung).

C-Aufruf

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
MQCXP  ChannelExitParms;   /* Channel exit parameter block */
MQCD   ChannelDefinition;  /* Channel definition */
```

Aufruf in COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
```

RPG-Aufrufe (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                   CALLP     exitname(MQCXP : MQCD)
```

Die Prototypdefinition für den Aufruf ist:

```
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
```

System/390 -Assembleraufruf

```
        CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

Die an den Exit übergebenen Parameter werden wie folgt deklariert:

```
CHANNELEXITPARMS   CMQCXPA  ,  Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,  Channel definition
```

Hinweise zur Verwendung

1. Die vom Kanalexit ausgeführte Funktion wird vom Exit-Provider definiert. Der Exit muss jedoch den hier und im zugehörigen Steuerblock (MQCXP) definierten Regeln entsprechen.
2. Der an den Exit für die automatische Kanaldefinition übergebene Parameter *ChannelExitParms* hat eine MQCXP-Struktur. Die übergebene MQCXP-Version richtet sich nach der Umgebung, in der der Exit ausgeführt wird. Ausführliche Informationen finden Sie in der Beschreibung des Felds *Version* unter „MQCXP - Kanalexitparameter” auf Seite 1100.
3. Der an den Exit für die automatische Kanaldefinition übergebene Parameter *ChannelDefinition* hat eine MQCD-Struktur. Die übergebene MQCD-Version richtet sich nach der Umgebung, in der der Exit ausgeführt wird. Ausführliche Informationen finden Sie in der Beschreibung des Felds *Version* unter „MQCD - Kanaldefinition” auf Seite 1058.

MQXWAIT - Wartezeit in Exit

Der MQXWAIT-Aufruf wartet auf ein stattfindendes Ereignis. Er kann nur von einem Kanalexit unter z/OS verwendet werden.

Die Verwendung des MQXWAIT-Aufrufs hilft, Leistungsprobleme zu vermeiden, die andernfalls auftreten könnten, wenn eine Aktion eines Kanalexits eine Wartezeit verursacht. Das Ereignis, auf das MQXWAIT wartet, wird von einem MVS-Ereignissteuerblock (ECB) signalisiert. Der Ereignissteuerblock wird in der Beschreibung "MQXWD-Steuerblock" näher erläutert.

Syntax

MQXWAIT *(Hconn, WaitDesc, CompCode, Reason)*

Parameter

Der MQXWAIT-Aufruf hat folgende Parameter:

Hconn (MQHCONN) - Eingabe

Verbindungskennung.

Diese Kennung steht für die Verbindung mit dem Warteschlangenmanager.Der Wert von *Hconn* wurde von einem vorherigen MQCONN-Aufruf zurückgegeben, der im gleichen oder früheren Aufruf des Exits aufgerufen wurde.

WaitDesc (MQXWD) - Ein-/Ausgabe

Wartezeitdeskriptor.

Dieser Parameter beschreibt das Ereignis, auf das gewartet wird.Ausführliche Informationen zu den Feldern in dieser Struktur finden Sie unter „MQXWD - Exit-Wait-Deskriptor” auf Seite 1114.

CompCode (MQLONG) - Ausgabe

Beendigungscode.

Hierbei handelt es sich um einen der folgenden Codes:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ausgabe

Ursachencode zur Qualifikation von *CompCode*.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter nicht verfügbar.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Optionen ungültig oder nicht konsistent.

MQRC_XWAIT_CANCELED
(2107, X'83B') Aufruf MQXWAIT wurde abgebrochen.

MQRC_XWAIT_ERROR
(2108, X'83C') Aufruf von MQXWAIT nicht gültig.

C-Aufruf

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONN  Hconn;     /* Connection handle */
MQXWD    WaitDesc;  /* Wait descriptor */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

System/390 -Assembleraufruf

```
         CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Deklarieren Sie die Parameter wie folgt:

```
HCONN     DS       F  Connection handle
WAITDESC  CMQXWDA  ,  Wait descriptor
COMPCODE  DS       F  Completion code
REASON    DS       F  Reason code qualifying COMPCODE
```

MQCD - Kanaldefinition

Die MQCD-Struktur enthält die Parameter, die die Ausführung eines Kanals steuern. Sie wird an jeden Kanalexit übergeben, der von einem Nachrichtenkanalagenten (MCA) aufgerufen wird.

Weitere Informationen zu Kanalexits finden Sie unter „MQ_CHANNEL_EXIT - Kanalexit” auf Seite 1051. Die Beschreibungen in diesem Abschnitt gelten sowohl für Nachrichtenkanäle als auch für MQI-Kanäle.

Felder mit dem Exitnamen

Beim Aufruf eines Exits enthält das jeweils zutreffende Feld der Felder *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit* und *MsgRetryExit* den Namen des aufgerufenen Exits. Die Bedeutung der Namen in diesen Feldern variiert je nach Umgebung, in der der Nachrichtenkanalagent ausgeführt wird. Wenn nicht anders angegeben, wird der Name innerhalb des Felds links ausgerichtet, wobei innerhalb des Namens keine Leerzeichen eingefügt werden. Am Ende des Namens wird er mit Leerzeichen auf die Länge des Felds aufgefüllt. In den nachfolgenden Beschreibungen kennzeichnen eckige Klammern ([ ]) optionale Informationen:

UNIX-Systeme
Der Exitname ist der Name eines dynamisch ladbaren Moduls bzw. einer Bibliothek, dem als Suffix der Name einer Funktion angefügt wird, die sich in dieser Bibliothek befindet. Der Funktionsname muss in Klammern eingeschlossen sein. Dem Bibliotheksnamen kann optional ein Verzeichnispfad vorangestellt werden:

```
[path]library(function)
```

Der Name darf maximal 128 Zeichen lang sein.

z/OS
Der Exitname ist der Name eines Lademoduls, der für Spezifikationen im EP-Parameter des LINK- oder LOAD-Makros verwendet werden kann. Der Name darf maximal acht Zeichen lang sein.

Windows
Der Exitname ist der Name einer DLL (Dynamic-Link Library), dem als Suffix der Name einer Funktion angefügt wird, die sich in dieser Bibliothek befindet. Der Funktionsname muss in Klammern eingeschlossen sein. Dem Bibliotheksnamen kann optional der Verzeichnispfad und das Laufwerk vorangestellt werden:

```
[d:][path]library(function)
```

Der Name darf maximal 128 Zeichen lang sein.

IBM i
Der Exitname ist ein 10 Byte großer Programmname gefolgt von einem 10 Byte großen Bibliotheksnamen. Falls die Namen kleiner als 10 Byte sind, werden sie mit Leerzeichen auf 10 Byte aufgefüllt. Der Bibliotheksname kann *LIBL sein, es sei denn, ein Exit für die automatische Kanaldefinition wird aufgerufen, in welchem Fall ein vollständig qualifizierter Name erforderlich ist.

Ändern von MQCD-Feldern in einem Kanalexit

Die Felder im MQCD können vom Kanalexit geändert werden. Der geänderte Wert verbleibt im MQCD und wird an alle verbleibenden Exits einer Exitkette sowie an alle Datenübertragungen übergeben, die die Kanalinstanz gemeinsam nutzen. Der geänderte MQCD wird zudem vom Nachrichtenkanalagenten für seine normalen Prozesse während der weiteren Lebensdauer des Kanals verwendet.

Die folgenden MQCD-Felder dürfen nicht vom Exit geändert werden:

- ChannelName
- ChannelType
- StrucLength
- Version

Zugehörige Verweise
„Felder” auf Seite 1060

In diesem Kapitel werden alle Felder der MQCD-Struktur aufgeführt und beschrieben.

„Deklaration in Programmiersprache C” auf Seite 1087
Bei dieser Deklaration handelt es sich um die C-Deklaration für die MQCD-Struktur.

„COBOL-Delaration” auf Seite 1089
Bei dieser Deklaration handelt es sich um die COBOL-Deklaration für die MQCD-Struktur.

„Deklaration in RPG (ILE)” auf Seite 1091
Bei dieser Deklaration handelt es sich um die RPG-Deklaration für die MQCD-Struktur.

„Deklaration in System/390 Assembler” auf Seite 1094
Bei dieser Deklaration handelt es sich um die System/390-Assemblerdeklaration für die MQCD-Struktur.

„Deklaration in Visual Basic” auf Seite 1095
Bei dieser Deklaration handelt es sich um die Visual Basic-Deklaration der MQCD-Struktur.

„Ändern von MQCD-Feldern in einem Kanalexit” auf Seite 1097
Die Felder im MQCD können vom Kanalexit geändert werden. Diese Änderungen werden jedoch nur unter den nachfolgend aufgeführten Bedingungen berücksichtigt.

Felder

In diesem Kapitel werden alle Felder der MQCD-Struktur aufgeführt und beschrieben.

BatchHeartbeat (MQLONG)
Dieses Feld gibt das Zeitintervall an, mit dem ein Überwachungssignal für den Stapelbetrieb des Kanals ausgelöst wird.

Über den Austausch von Überwachungssignalen für den Stapelbetrieb können Senderkanäle bestimmen, ob die ferne Kanalinstanz noch aktiv ist, bevor sie sie auflösen. Ein Überwachungssignal für den Stapelbetrieb tritt auf, wenn ein Senderkanal nicht innerhalb des angegebenen Zeitintervalls mit der fernen Kanalinstanz kommuniziert hat.

Der Wert liegt zwischen 0 bis 999999, die Einheiten sind in Millisekunden angegeben. Der Wert 0 zeigt an, dass kein Austausch von Überwachungssignalen für den Stapelbetrieb aktiviert ist.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

BatchInterval (MQLONG)
Dieses Feld gibt die ungefähre Dauer in Millisekunden an, für die ein Kanal einen Stapel offen hält. Bei weniger als *BatchSize* wurden Nachrichten im aktuellen Stapel übertragen.

Wenn *BatchInterval* größer ist als null, wird der Stapel von dem Ereignis beendet, das zuerst eintrifft:

- die in *BatchSize* angegebene Anzahl Nachrichten wurde gesendet oder
- *BatchInterval* Millisekunden sind seit dem Start des Stapels vergangen.

Wenn *BatchInterval* null ist, wird der Stapel von dem Ereignis beendet, das zuerst eintritt:

- die in *BatchSize* angegebene Anzahl Nachrichten wurde gesendet oder
- die Übertragungswarteschlange ist leer.

BatchInterval muss einen Wert zwischen 0 und 999999999 haben.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

BatchSize (MQLONG)
Dieses Feld gibt die maximale Anzahl an Nachrichten an, die über einen Kanal gesendet werden können, bevor dieser synchronisiert wird.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_SVRCONN oder MQCHT_CLNTCONN ohne Bedeutung.

ChannelMonitoring (MQLONG)
Dieses Feld gibt die aktuelle Stufe der Erfassung von Überwachungsdaten für den Kanal an.

Dieses Feld ist für Kanäle mit dem Kanaltyp MQCHT_CLNTCONN ohne Bedeutung.

Folgende Werte sind möglich:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

ChannelName (MQCHAR20)
Dieses Feld gibt den Namen der Kanaldefinition an.

Damit eine Kommunikation möglich ist, muss auf dem fernen System eine Kanaldefinition mit demselben Namen vorhanden sein.

Der Name darf nur die folgenden Zeichen enthalten:

- Großbuchstaben A-Z
- a-z in Kleinbuchstaben
- Zahlen 0-9
- Punkt (.)
- Schrägstrich (/)
- Unterstrich (_)
- Prozentzeichen (%)

Nach rechts wird er mit Leerzeichen aufgefüllt. Führende oder eingebettete Leerzeichen sind nicht erlaubt.

Die Länge dieses Felds ist durch MQ_CHANNEL_NAME_LENGTH vorgegeben.

ChannelStatistics (MQLONG)
Dieses Feld gibt die aktuelle Stufe der Erfassung statistischer Daten für den Kanal an.

Dieses Feld ist für Kanäle mit dem Kanaltyp MQCHT_CLNTCONN ohne Bedeutung.

Folgende Werte sind möglich:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

ChannelType (MQLONG)
Dieses Feld gibt den Kanaltyp an.

Folgende Werte sind möglich:

MQCHT_SENDER
Sender

MQCHT_SERVER
Server.

MQCHT_RECEIVER
Empfänger.

MQCHT_REQUESTER
Requester

MQCHT_CLNTCONN
Clientverbindung.

MQCHT_SVRCONN
Serververbindung (zur Verwendung durch Clients).

MQCHT_CLUSSDR
Clustersender.

MQCHT_CLUSRCVR
Clusterempfänger.

ClientChannelWeight (MQLONG)
Dieses Feld gibt eine Gewichtung an, durch die gesteuert wird, welche Clientverbindungskanaldefinition verwendet wird.

Mit dem Attribut ClientChannelWeight können Clientkanaldefinitionen auf der Basis ihrer Gewichtung ausgewählt werden, wenn mehrere geeignete Definitionen zur Verfügung steht. Wenn ein Client MQCONN ausgibt, um eine Verbindung zu einer Warteschlangenmanagergruppe anzufordern, und dabei einen mit einem Stern beginnenden Warteschlangenmanagernamen angibt, wodurch die Clientgewichtung für mehrere Warteschlangenmanager ermöglicht wird, wird die zu verwendende Definition auf der Basis der Gewichtung ausgewählt, wenn die Definitionstabelle für Clientkanäle (CCDT) mehrere geeignete Kanaldefinitionen enthält. Dabei werden gültige Definitionen des Typs CLNTWGHT(0) in alphabetischer Reihenfolge zuerst ausgewählt.

Geben Sie einen Wert im Bereich von 0 bis 99 an. Der Standardwert ist 0.

Der Wert 0 gibt an, dass kein Lastausgleich erfolgt und gültige Definitionen in alphabetischer Reihenfolge ausgewählt werden. Wenn der Lastausgleich aktiviert werden soll, wählen Sie einen Wert im Bereich von 1 bis 99 aus, wobei 1 der niedrigsten und 99 der höchsten Gewichtung entspricht. Die Aufteilung der Nachrichten zwischen zwei oder mehreren Kanälen mit einer Gewichtung ungleich null erfolgt proportional zum Verhältnis dieser Gewichtungen. Es werden beispielsweise drei Kanäle mit den CLNTWGHT-Werten 2, 4 und 14 zu rund 10 %, 20 % und 70 % der Zeit ausgewählt. Diese Verteilung ist nicht garantiert.

Dieses Attribut ist nur für den Kanaltyp Clientverbindungskanal gültig.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_9 ist.

ClusterPtr (MQPTR)
Dieses Feld gibt die Adresse einer Liste mit Clusternamen an.

Wenn *ClustersDefined* größer ist als null, ist diese Adresse die Adresse einer Liste mit Clusternamen.Der Kanal gehört zu jedem in der Liste aufgeführten Cluster.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_5.

ClustersDefined (MQLONG)
Dieses Feld gibt die Anzahl der Cluster an, zu denen der Kanal gehört.

Dieses Feld enthält die Anzahl der Clusternamen, auf die *ClusterPtr* verweist.Der Wert dieses Felds ist null oder größer.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)
Dieses Feld gibt die Kanalpriorität der Clusterauslastung an.

Der Auswahlalgorithmus des Workload-Managers wählt aus der Gruppe der Zieladressen, die basierend auf dem Rang ausgewählt wurden, eine Zieladresse mit der höchsten Priorität aus. Wenn es zwei mögliche Zielwarteschlangenmanager gibt, kann mithilfe dieses Attributs festgelegt werden, dass ein Warteschlangenmanager auf den anderen Warteschlangenmanager ausweicht. Alle Nachrichten werden an den Warteschlangenmanager mit der höchsten Priorität gesendet, bis diese endet. Danach gehen die Nachrichten an den Warteschlangenmanager mit der nächsthöheren Priorität.

Der Wert liegt im Bereich von 0 bis 9. Der Standardwert ist 0.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

Weitere Informationen finden Sie im Abschnitt Warteschlangenmanagercluster konfigurieren.

CLWLChannelRank (MQLONG)
Dieses Feld gibt den Kanalrang der Clusterauslastung an.

Der Auswahlalgorithmus des Workload-Managers wählt eine Zieladresse mit dem höchsten Rang aus. Wenn es sich bei der Zieladresse um einen Warteschlangenmanager in einem anderen Cluster handelt, können Sie den Rang eines temporären Gateway-Warteschlangenmanagers (am Schnittpunkt zu benachbarten Clustern) festlegen, damit der Auswahlalgorithmus einen Zielwarteschlangenmanager auswählt, der näher an der Zieladresse liegt.

Der Wert liegt im Bereich von 0 bis 9. Der Standardwert ist 0.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

Weitere Informationen finden Sie unter Warteschlangenmanagercluster konfigurieren .

CLWLChannelWeight (MQLONG)
Dieses Feld gibt die Kanalgewichtung der Clusterauslastung an.

Kanalgewichtung für Clusterauslastung.

Der Auswahlalgorithmus des Workload-Managers verwendet das Gewichtungsattribut ("weight") des Kanals, um die Auswahl der Zieladresse ungleich zu verteilen, damit mehr Nachrichten an eine bestimmte Maschine gesendet werden können. Sie können beispielsweise einem Kanal auf einem großen UNIX-Server eine höhere Gewichtung geben als einem anderen Kanal auf einem kleinen Desktop-PC, damit der Auswahlalgorithmus den UNIX-Server öfter auswählt als den PC.

Der Wert liegt zwischen 1 bis 99. Der Standardwert lautet 50.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

Weitere Informationen finden Sie im Abschnitt Warteschlangenmanagercluster konfigurieren.

ConnectionAffinity (MQLONG)
Dieses Feld gibt an, ob Clientanwendungen, bei denen mehrfach Verbindungen mit dem gleichen Warteschlangenmanagernamen hergestellt werden, denselben Clientkanal verwenden.

Verwenden Sie dieses Attribut, wenn mehrere gültige Kanaldefinitionen verfügbar sind.

Folgende Werte sind möglich:

MQCAFTY_PREFERRED
Die erste Verbindung eines Prozesses, die eine Definitionstabelle für Clientkanäle (CCDT) liest, erstellt basierend auf der Gewichtung eine Liste gültiger Definitionen, in der die Definitionen mit der Gewichtung CLNTWGHT(0) jeweils in alphabetischer Reihenfolge zuerst aufgeführt sind. Bei jeder Verbindung des Prozesses wird versucht, die Verbindung über die erste Definition der Liste herzustellen. Wenn eine Verbindung nicht erfolgreich ist, wird die nächste Definition verwendet. Erfolglose Definitionen mit CLNTWGHT-Werten ungleich 0 werden an das Ende der Liste verschoben. CLNTWGHT(0)-Definitionen verbleiben am Anfang der Liste und werden für jede Verbindung zuerst ausgewählt.

Jeder Clientprozess mit demselben Hostnamen erstellt immer dieselbe Liste.

Bei Clientanwendungen, die in C, C++ oder im .NET-Programmierframework geschrieben wurden (einschließlich vollständig verwalteter .NET-Clientanwendungen), wird die Liste aktualisiert, wenn die Definitionstabelle für Clientkanäle (CCDT) seit Erstellung der Liste geändert wurde.

Dies ist der Standardwert.

MQCAFTY_NONE
Die erste Verbindung eines Prozesses, die eine CCDT liest, erstellt eine Liste gültiger Definitionen. Alle Verbindungen in einem Prozess wählen eine gültige Definition auf der Basis der Gewichtung aus, wobei alle gültigen CLNTWGHT(0)-Definitionen zuerst und in alphabetischer Reihenfolge ausgewählt werden.

Bei Clientanwendungen, die in C, C++ oder im .NET-Programmierframework geschrieben wurden (einschließlich vollständig verwalteter .NET-Clientanwendungen), wird die Liste aktualisiert, wenn die Definitionstabelle für Clientkanäle (CCDT) seit Erstellung der Liste geändert wurde.

Dieses Attribut ist nur für den Kanaltyp Clientverbindungskanal gültig.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_9 ist.

ConnectionName (MQCHAR264)
Dieses Feld gibt den Verbindungsnamen des Kanals an.

Bei Clusterempfängerkanal bezieht sich CONNAME (wenn angegeben) auf den lokalen Warteschlangenmanager, bei den anderen Kanälen auf den Ziel-Warteschlangenmanager. Der Wert, den Sie angeben, hängt vom Übertragungsprotokoll (*TransportType*) ab, das verwendet werden soll:

- Bei MQXPT_LU62 ist es der vollständig qualifizierte Name der Partner-LU.
- Bei MQXPT_NETBIOS ist es der auf dem fernen System definierte NetBIOS-Name.
- Bei MQXPT_TCP ist es entweder der Hostname oder die Netzadresse des fernen Systems (in IPv4 angegeben im Format mit Trennzeichen, in IPv6 im Hexadezimalformat) oder bei Clusterempfängerkanälen das ferne System.
- Bei MQXPT_SPX ist eine SPX-Adresse, bestehend aus einer Netzadresse mit 4 Byte oder einer Knotenadresse mit 6 Byte und einer Socketadresse mit 2 Byte.

Beim Definieren eines Kanals ist dieses Feld für Kanäle vom *ChannelType* MQCHT_SVRCONN oder MQCHT_RECEIVER ohne Bedeutung. Wenn die Kanaldefinition jedoch an den Exit übergeben wird, enthält dieses Feld die Adresse des Partners, und zwar unabhängig vom Kanaltyp.

Die Länge dieses Feldes ist durch MQ_CONN_NAME_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

DataConversion (MQLONG)
Gibt an, ob der Agent des sendenden Nachrichtenkanals versuchen soll, die Nachrichtendaten der Anwendung zu konvertieren, wenn der Agent des empfangenden Nachrichtenkanals diese Konvertierung nicht durchführen kann.

Dieses Feld gilt nur für Nachrichten, bei denen es sich nicht um Segmente logischer Nachrichten handelt. Der Nachrichtenkanalagent versucht niemals, Nachrichten zu konvertieren, bei denen es sich um Segmente handelt.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung. Folgende Werte sind möglich:

MQCDC_SENDER_CONVERSION
Konvertierung durch den Sender.

MQCDC_NO_SENDER_CONVERSION
Keine Konvertierung durch den Sender.

DefReconnect (MQLONG)
Das Kanalattribut `DefReconnect` legt den Standardwert für das Verbindungswiederholungsattribut eines Clientverbindungskanals fest.

Die Standardoption für automatische Clientverbindungswiederholung. Sie können einen IBM WebSphere MQ MQI client so konfigurieren, dass er die Verbindung mit einer Clientanwendung automatisch wiederholt. Der IBM WebSphere MQ MQI client versucht nach einem Verbindungsfehler, die Verbindung mit einem Warteschlangenmanager zu wiederholen. Er versucht dies, ohne dass der Anwendungsclient den MQI-Aufruf `MQCONN` oder `MQCONNX` ausgibt.

Verbindungswiederholung ist eine `MQCONNX`-Option. Indem Sie das Kanalattribut `DefReconnect` verwenden, können Sie Verbindungswiederholungsverhalten zu bestehenden Anwendungen hinzufügen, die `MQCONN` verwenden. Sie können auch das Verbindungswiederholungsverhalten von Anwendungen ändern, die `MQCONNX` verwenden.

Sie können auch den Wert `DefRecon` aus der Liste `mqclient.ini` so setzen, dass das Verbindungswiederholungsverhalten festgelegt oder geändert wird. Der Wert `DefRecon` aus der Datei `mqclient.ini` hat Vorrang vor dem Kanalattribut `DefReconnect`.

Syntax

```
DefReconnect( MQRCN_NO|MQRCN_YES|MQRCN_Q_MGR|
MQRCN_DISABLED)
```

Parameter

MQRCN_NO
`MQRCN_NO` ist der Standardwert.

Sofern nicht von `MQCONNX`überschrieben, wird die Clientverbindung nicht automatisch wiederhergestellt.

MQRCN_YES
Wenn nicht durch `MQCONNX`überschrieben, stellt der Client die Verbindung automatisch wieder her.

MQRCN_Q_MGR
Wenn nicht durch `MQCONNX`überschrieben, stellt der Client die Verbindung automatisch wieder her, aber nur mit demselben Warteschlangenmanager. Die Option `QMGR` hat dieselbe Wirkung wie `MQCNO_RECONNECT_Q_MGR`.

MQRCN_DISABLED
Die Verbindungswiederholung ist inaktiviert, auch wenn sie vom Clientprogramm mit dem MQI-Aufruf `MQCONNX` angefordert wird.

Die automatische Clientverbindungswiederholung wird von IBM WebSphere MQ -Klassen für Java nicht unterstützt.

Tabelle 592. Automatische Verbindungswiederholung hängt von den in der Anwendung und in der Kanaldefinition gesetzten Werten ab.

DefReconnect	**In der Anwendung festgelegte Verbindungswiederholungsoptionen**			
	`MQCNO_RECONNECT`	`MQCNO_RECONNECT_Q_MGR`	`MQCNO_RECONNECT_AS_DEF`	`MQCNO_RECONNECT_DISABLED`
`MQRCN_NO`	YES	QMGR	Nein	Nein

Tabelle 592. Automatische Verbindungswiederholung hängt von den in der Anwendung und in der Kanaldefinition gesetzten Werten ab. (Forts.)				
DefReconnect	**In der Anwendung festgelegte Verbindungswiederholungsoptionen**			
MQRCN_YES	YES	QMGR	YES	Nein
MQRCN_Q_MGR	YES	QMGR	QMGR	Nein
MQRCN_DISABLED	Nein	Nein	Nein	Nein

Zugehörige Konzepte

Automatische Clientverbindungswiederholung

Kanal- und Clientverbindungswiederholung

Zeilengruppe 'CHANNELS' in der Clientkonfigurationsdatei

Zugehörige Verweise

Verbindungsoptionen

Optionen, mit denen die Aktion des MQCONNX-Aufrufs gesteuert wird.

Desc (MQCHAR64)
Dieses Feld kann zur beschreibenden Erläuterung verwendet werden.

Der Inhalt des Felds hat für Nachrichtenkanalagenten keine Bedeutung. Es darf jedoch nur anzeigbare Zeichen und keinerlei Nullzeichen enthalten. Das Feld darf keine Nullzeichen enthalten; ggf. wird es rechts mit Leerzeichen aufgefüllt. In einer DBCS-Installation kann das Feld DBCS-Zeichen enthalten (die maximale Länge beträgt 64 Byte).

Anmerkung: Wenn dieses Feld Zeichen enthält, die nicht im Zeichensatz des Warteschlangenmanagers enthalten sind (definiert durch das Warteschlangenmanagerattribut *CodedCharSetId*), werden diese Zeichen unter Umständen falsch übersetzt, wenn das Feld an einen anderen Warteschlangenmanager gesendet wird.

Die Länge dieses Felds ist durch MQ_CHANNEL_DESC_LENGTH vorgegeben.

DiscInterval (MQLONG)
Dieses Feld gibt die maximale Zeit in Sekunden an, die der Kanal auf eine Nachricht in der Übertragungswarteschlange wartet, bevor der Kanal beendet wird.

Es gibt also das Intervall zur Verbindungstrennung an.

Bei einem Nullwert wartet der Nachrichtenkanalagent auf unbestimmte Zeit.

Bei Serververbindungskanälen, die das TCP-Protokoll verwenden, gibt das Intervall die Zeit in Sekunden an, bis die Verbindung bei Inaktivität des Clients getrennt wird. Wenn eine Serververbindung für diese Dauer keine Kommunikation von ihrem Partnerclient erhalten hat, beendet sie die Verbindung. Das Intervall bei inaktiver Serververbindung gilt nur zwischen WebSphere MQ-API-Aufrufen von einem Client. Während eines lange laufenden MQGET mit Warteaufruf wird also kein Client getrennt.

Bei Serververbindungskanälen, die ein anderes Protokoll als TCP verwenden, wird dieses Attribut ignoriert.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR oder MQCHT_SVRCONN von Bedeutung.

ExitDataLength (MQLONG)
Dieses Feld gibt die Länge in Byte aller Benutzerdatenelemente in den Listen der Datenelemente von Exitbenutzern an, die von den Feldern *MsgUserDataPtr*, *SendUserDataPtr* und *ReceiveUserDataPtr* angesprochen werden.

Diese Länge entspricht nicht zwangsläufig der Länge MQ_EXIT_DATA_LENGTH.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

ExitNameLength (MQLONG)
Dieses Feld gibt die Länge in Byte jedes Namens in den Listen der Exitnamen an, die von den Feldern *MsgExitPtr*, *SendExitPtr* und *ReceiveExitPtr* angesprochen werden.

Diese Länge entspricht nicht zwangsläufig der Länge MQ_EXIT_NAME_LENGTH.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

HdrCompList [2] (MQLONG)
Dieses Feld gibt die Liste mit den Komprimierungsverfahren für Headerdaten an, die vom Kanal unterstützt werden.

Die Liste enthält mindestens einen der folgenden Werte:

MQCOMPRESS_NONE
Es werden keine Headerdaten komprimiert.

MQCOMPRESS_SYSTEM
Headerdaten werden komprimiert.

Nicht in der Gruppe verwendete Werte sind auf MQCOMPRESS_NOT_AVAILABLE gesetzt.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

HeartbeatInterval (MQLONG)
Dieses Feld gibt die Sekunden zwischen den Austauschen von Überwachungssignalen an.

Die Interpretation dieses Felds richtet sich wie folgt nach dem Kanaltyp:

- Beim Kanaltyp MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR gibt dieses Feld die Sekunden zwischen den Austauschen von Überwachungssignalen an, die vom sendenden Nachrichtenkanalagenten übertragen werden, wenn sich keine Nachrichten in der Übertragungswarteschlange befinden. Dies gibt dem empfangenden Nachrichtenkanalagent die Gelegenheit, den Kanal in den Quiescemodus zu versetzen. Um von Nutzen zu sein, muss *HeartbeatInterval* kleiner sein als *DiscInterval*.
- Wenn bei den Kanaltypen MQCHT_CLNTCONN oder MQCHT_SVRCONN das MQCD-Feld "SharingConversations" auf Null gesetzt ist, gibt dieses Feld die Sekunden zwischen den Austauschen von Überwachungssignalen an, die vom Nachrichtenkanalagent des Servers übergeben werden, wenn der Nachrichtenkanalagent im Namen einer Clientanwendung einen MQGET-Aufruf mit der Option MQGMO_WAIT ausgegeben hat. Damit kann der Nachrichtenkanalagent des Servers Situationen verarbeiten, in denen die Clientverbindung während eines MQGET-Aufrufs unter Angabe der Option MQGMO_WAIT unterbrochen wird.
- Wenn bei den Kanaltypen MQCHT_CLNTCONN oder MQCHT_SVRCONN das MQCD-Feld "SharingConversations" auf einen Wert ungleich null gesetzt ist, gibt dieses Feld die Sekunden zwischen den Austauschen von Überwachungssignalen an, wenn keine Datenflüsse gesendet oder empfangen wurden. So kann der Kanal effizient in den Quiescemodus versetzt werden.

Der Wert liegt zwischen 0 und 999999. Der verwendete Wert ist der größere der Werte, die auf der Sende- und Empfangsseite festgelegt sind, solange an keiner der Seiten ein Wert von 0 angegeben ist, der den Austausch von Überwachungssignalen verhindert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

KeepAliveInterval (MQLONG)
Dieses Feld gibt den Wert für das Keepalive-Timing für diesen Kanal an, der an den Kommunikationsstack übergeben wird.

Der Wert ist gültig für TCP/IP- und SPX-Kommunikationsprotokolle, obwohl nicht alle Implementierungen diesen Parameter unterstützen.

Der Wert liegt zwischen 0 bis 99999, die Einheiten sind in Sekunden angegeben. Der Wert Null zeigt an, dass kein Kanal-Keepalive aktiviert ist, obwohl Keepalive dennoch auftreten kann, wenn TCP/IP-Keepalive (statt Kanal-Keepalive) aktiviert ist. Folgender Sonderwert ist ebenfalls gültig:

MQKAI_AUTO
Automatisch.

Dieser Wert zeigt an, dass das Keepalive-Intervall wie folgt aus dem verhandelten Intervall der Überwachungssignale berechnet wird:

- Ist für das verhandelte Intervall für Überwachungssignale ein Wert größer als 0 angegeben, wird als Keepalive-Intervall das Intervall der Überwachungssignale plus 60 Sekunden verwendet.
- Hat das verhandelte Intervall für Überwachungssignale den Wert 0, beträgt der Wert des verwendeten Keepalive-Intervalls ebenfalls 0.

- Unter z/OS tritt TCP/IP-Keepalive auf, wenn TCPKEEP(YES) im Warteschlangenmanagerobjekt angegeben ist.
- In anderen Umgebungen tritt TCP/IP-Keepalive auf, wenn der Parameter KEEPALIVE=YES in der TCP-Zeilengruppe der Konfigurationsdatei für die verteilte Steuerung von Warteschlangen angegeben wurde.

Dieses Feld ist nur für Kanäle mit dem *TransportType* MQXPT_TCP oder MQXPT_SPX gültig.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

LocalAddress (MQCHAR48)
Dieses Feld gibt die lokale TCP/IP-Adresse an, die für den Kanal für abgehende Kommunikation definiert ist.

Dieses Feld ist leer, wenn keine bestimmte Adresse für abgehende Kommunikation definiert ist.Optional kann die Adresse eine Portnummer bzw. einen bestimmten Bereich von Portnummern beinhalten. Das Format dieser Adresse lautet wie folgt:

```
[ip-addr][(low-port[,high-port])]
```

Dabei stehen eckige Klammern ([ ]) für optionale Informationen, `ip-addr` ist eine IPv4-Adresse (in Schreibweise mit Trennzeichen), eine IPv6-Adresse (in Hexadezimalschreibweise) oder eine Adresse in alphanumerischer Schreibweise. Für `low-port` und `high-port` werden in Klammern gesetzte Portnummern angegeben. Alle Angaben sind optional.

Eine bestimmte IP-Adresse, ein bestimmter Port oder Portbereich für die abgehende Kommunikation ist nützlich bei Wiederherstellungsszenarios, wenn ein Kanal in einem anderen TCP/IP-Stapel neu gestartet wird.

Auch wenn das Format von *LocalAddress* dem Format von *ConnectionName* ähnelt, sind diese beiden nicht zu verwechseln. *LocalAddress* gibt die Merkmale der lokalen Kommunikation an, während *ConnectionName* angibt, wie ein ferner Warteschlangenmanager erreicht werden kann.

Dieses Feld ist nur für Kanäle mit dem *TransportType* MQXPT_TCP und dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Die Länge dieses Felds ist durch MQ_LOCAL_ADDRESS_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)
Dieses Feld gibt die Länge (in Byte) der vollständigen Benutzer-ID des Nachrichtenkanalagenten an, auf die *LongMCAUserIdPtr* verweist.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_CLNTCONN ohne Bedeutung.

Dies ist ein Ein-/Ausgabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)
Dieses Feld gibt die Adresse der langen Benutzer-ID des Nachrichtenkanalagenten an.

Wenn *LongMCAUserIdLength* größer ist als null, ist dieses Feld die Adresse der vollständigen Benutzer-ID des Nachrichtenkanalagenten.Die Länge der vollständigen ID wird durch *LongMCAUserIdLength* vorgegeben. Die ersten 12 Byte der Benutzer-ID des Nachrichtenkanalagenten sind auch im Feld *MCAUserIdentifier* enthalten.

Ausführliche Informationen zur Benutzer-ID des Nachrichtenkanalagenten finden Sie in der Beschreibung des *MCAUserIdentifier*-Felds.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN oder MQCHT_CLUSSDR ohne Bedeutung.

Dies ist ein Ein-/Ausgabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

LongRemoteUserIdLength (MQLONG)
Dieses Feld gibt die Länge (in Byte) der vollständigen Remotebenutzer-ID an, auf die *LongRemoteUserIdPtr* verweist.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLNTCONN oder MQCHT_SVRCONN von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)
Dieses Feld gibt die Adresse der langen Remotebenutzer-ID an.

Wenn *LongRemoteUserIdLength* größer ist als null, ist dieses Flag die Adresse der vollständigen Remotebenutzer-ID.Die Länge der vollständigen ID wird durch *LongRemoteUserIdLength* vorgegeben. Die ersten 12 Byte der Remotebenutzer-ID sind auch im Feld *RemoteUserIdentifier* enthalten.

Ausführliche Informationen zur Remotebenutzer-ID finden Sie in der Beschreibung des *RemoteUserIdentifier*-Felds.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLNTCONN oder MQCHT_SVRCONN von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

LongRetryCount (MQLONG)
Dieses Feld gibt den Zähler an, der verwendet werden soll, nachdem der durch *ShortRetryCount* festgelegte Zähler ausgeschöpft ist.

Dieses Feld gibt die maximale Anzahl der erneuten Verbindungsversuche mit dem fernen System an. Die Intervalle werden vom Parameter *LongRetryInterval* angegeben, bevor dem Operator ein Fehler gemeldet wird.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

LongRetryInterval (MQLONG)
Dieses Feld gibt die maximale Anzahl an Sekunden an, nach denen erneut versucht wird, eine Verbindung zum fernen System herzustellen.

Das Intervall zwischen den Verbindungsversuchen kann erhöht werden, wenn der Kanal abwarten muss, bis er aktiv ist.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

MaxInstances (MQLONG)
Dieses Feld gibt die maximale Anzahl simultaner Instanzen eines einzelnen Serververbindungskanals an, die gestartet werden können.

Dieses Feld wird nur für Serververbindungskanäle verwendet.

In dieses Feld kann ein Wert zwischen 0 und 999 999 999 eingegeben werden. Bei einem Wert von 0 (null) wird der Clientzugriff verhindert.

Der Standardwert für dieses Feld ist 999 999 999.

Wenn der in diesem Feld angegebene Wert kleiner ist als die Anzahl der derzeit aktiven Instanzen des Serververbindungskanals, sind diese aktiven Instanzen nicht betroffen. Allerdings können in diesem Fall neue Instanzen erst dann gestartet werden, wenn genügend aktive Instanzen beendet wurden, sodass die Anzahl der derzeit aktiven Instanzen unter dem Wert in diesem Feld liegt.

MaxInstancesPerClient (MQLONG)
Dieses Feld gibt die maximale Anzahl simultaner Instanzen eines einzelnen Serververbindungskanals an, die auf einem einzelnen Client gestartet werden können.

In diesem Zusammenhang werden Verbindungen, die von derselben Remotenetzwerkadresse stammen, als von demselben Client kommend betrachtet.

Dieses Feld wird nur für Serververbindungskanäle verwendet.

In dieses Feld kann ein Wert zwischen 0 und 999 999 999 eingegeben werden. Bei einem Wert von 0 (null) wird der Clientzugriff verhindert.

Der Standardwert für dieses Feld ist 999 999 999.

Wenn der in diesem Feld angegebene Wert kleiner ist als die Anzahl der derzeit aktiven Instanzen des Serververbindungskanals von einzelnen Clients, sind diese aktiven Instanzen nicht betroffen. Allerdings können in diesem Fall neue Instanzen von diesen Clients erst dann gestartet werden, wenn genügend aktive Instanzen beendet wurden, sodass die Anzahl der derzeit aktiven Instanzen des Clients, der versucht, eine neue Instanz zu starten, unter dem Wert in diesem Feld liegt.

MaxMsgLength (MQLONG)
Dieses Feld gibt die maximale Nachrichtenlänge an, die auf dem Kanal übertragen werden kann.

Diese Angabe wird mit dem Wert für den fernen Kanal verglichen. Der niedrigere der beiden Werte wird als maximale Länge übernommen.

MCAName (MQCHAR20)
Bei diesem Feld handelt es sich um ein reserviertes Feld.

Der Wert dieses Felds ist leer.

Die Länge dieses Felds ist durch MQ_MCA_NAME_LENGTH vorgegeben.

MCASecurityId (MQBYTE40)
Dieses Feld gibt die Sicherheits-ID des Nachrichtenkanalagenten an.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_CLNTCONN ohne Bedeutung.

Der folgende Sonderwert gibt an, dass keine Sicherheits-ID vorhanden ist:

MQSID_NONE
Es ist keine Sicherheits-ID angegeben.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQSID_NONE_ARRAY definiert. Diese Konstante hat den gleichen Wert wie die Konstante MQSID_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Ein-/Ausgabefeld für den Exit. Die Länge dieses Felds ist durch MQ_SECURITY_ID_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

MCAType (MQLONG)
Dieses Feld gibt den Programmtyp des Nachrichtenkanalagenten an.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Folgende Werte sind möglich:

MQMCAT_PROCESS
Prozess

Der Nachrichtenkanalagent läuft als separater Prozess.

MQMCAT_THREAD
Thread (IBM i, UNIX und Windows).

Der Nachrichtenkanalagent wird als separater Thread ausgeführt.

Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)
Dieses Feld gibt die Benutzer-ID des Nachrichtenkanalagenten (MCA) an.

Dieses Feld verwendet die ersten 12 Byte der Benutzer-ID des Nachrichtenkanalagenten und kann von einem Sicherheitsagenten gesetzt werden.

Zwei Felder enthalten die Benutzer-ID des Nachrichtenkanalagenten:

- *MCAUserIdentifier* enthält die ersten 12 Byte der Benutzer-ID des Nachrichtenkanalagenten und wird mit Leerzeichen aufgefüllt, wenn die ID kürzer ist als 12 Byte. *MCAUserIdentifier* kann leer sein.
- *LongMCAUserIdPtr* verweist auf die vollständige Benutzer-ID des Nachrichtenkanalagenten, die länger sein kann als 12 Byte. Ihre Länge wird durch *LongMCAUserIdLength* vorgegeben. Die vollständige ID enthält keine abschließenden Leerzeichen und endet nicht auf null. Ist die ID leer, hat *LongMCAUserIdLength* den Wert Null und der Wert von *LongMCAUserIdPtr* ist nicht definiert.

  Anmerkung: *LongMCAUserIdPtr* ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

Ist die Benutzer-ID des Nachrichtenkanalagenten nicht leer, gibt sie die Benutzer-ID für den Nachrichtenkanalagenten an, mit der dieser zum Zugriff auf WebSphere MQ-Ressourcen berechtigt ist. Wenn bei Verwendung der Kanaltypen MQCHT_REQUESTER, MQCHT_RECEIVER und MQCHT_CLUSRCVR für "PutAuthority" der Wert MQPA_DEFAULT festgelegt wird, ist dies die Benutzer-ID, die für Berechtigungsprüfungen der Put-Operation in Zielwarteschlangen verwendet wird.

Ist die Benutzer-ID des Nachrichtenkanalagenten leer, verwendet der Nachrichtenkanalagent seine Standard-Benutzer-ID.

Die Benutzer-ID des Nachrichtenkanalagenten kann von einem Sicherheitsexit so gesetzt werden, dass sie die Benutzer-ID anzeigt, die der Nachrichtenkanalagent verwenden muss. Der Exit kann entweder *MCAUserIdentifier* oder die Zeichenfolge ändern, auf die *LongMCAUserIdPtr* verweist. Wenn beide geändert werden, jedoch voneinander abweichen, verwendet der Nachrichtenkanalagent *LongMCAUserIdPtr* anstelle von *MCAUserIdentifier*. Wenn der Exit die Länge der durch *LongMCAUserIdPtr*adressierten Zeichenfolge ändert, muss *LongMCAUserIdLength* entsprechend festgelegt werden. Wenn der Exit die Länge der ID erhöht, muss er Speicher der benötigten Länge zuordnen, diesen Speicher auf die erforderliche ID setzen und die Adresse des Speichers in *LongMCAUserIdPtr* angeben. Der Exit ist für das Freimachen von Speicher verantwortlich, wenn der Exit später mit dem Grund MQXR_TERM aufgerufen wird.

Wenn bei Kanälen mit dem *ChannelType* MQCHT_SVRCONN die *MCAUserIdentifier* in der Kanaldefinition leer ist, wird jede vom Client übertragene Benutzer-ID dorthinein kopiert. Diese Benutzer-ID (nach Änderung durch den Sicherheitsexit auf dem Server) ist die ID, von der vorausgesetzt wird, dass die Clientanwendung darunter ausgeführt wird.

Die Benutzer-ID des Nachrichtenkanalagenten ist für Kanäle vom *ChannelType* MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN und MQCHT_CLUSSDR ohne Bedeutung.

Dies ist ein Ein-/Ausgabefeld für den Exit. Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

ModeName (MQCHAR8)
Dieses Feld gibt den LU 6.2-Modusnamen an.

Dieses Feld ist nur von Bedeutung, wenn das Übertragungsprotokoll (*TransportType*) MQXPT_LU62 und der *ChannelType* nicht MQCHT_SVRCONN oder MQCHT_RECEIVER lautet.

Dieses Feld ist immer leer. Die entsprechenden Informationen befinden sich stattdessen im Kommunikationsobjekt.

Die Länge dieses Felds ist durch MQ_MODE_NAME_LENGTH vorgegeben.

MsgCompList [16] (MQLONG)
Dieses Feld gibt die Liste mit den Komprimierungsverfahren für Nachrichtendaten an, die vom Kanal unterstützt werden.

Die Liste enthält mindestens einen der folgenden Werte:

MQCOMPRESS_NONE
Es werden keine Nachrichtendaten komprimiert.

MQCOMPRESS_RLE
Nachrichtendaten werden mittels Lauflängencodierung komprimiert.

MQCOMPRESS_ZLIBFAST
Die Komprimierung der Nachrichtendaten erfolgt unter Verwendung der ZLIB-Komprimierungstechnik. Dabei wird eine kurze Komprimierungszeit bevorzugt.

MQCOMPRESS_ZLIBHIGH
Die Komprimierung der Nachrichtendaten erfolgt unter Verwendung der ZLIB-Komprimierungstechnik. Dabei wird eine hohe Komprimierungsstufe bevorzugt.

Nicht in der Gruppe verwendete Werte sind auf MQCOMPRESS_NOT_AVAILABLE gesetzt.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_8.

MsgExit (MQCHARn)
Dieses Feld gibt den Namen des Kanalnachrichtenexits an.

Wenn der Name nicht leer ist, wird der Exit zu folgenden Zeiten aufgerufen:

- Sofort nachdem eine Nachricht aus der Übertragungswarteschlange (Sender oder Server) abgerufen wurde oder unmittelbar bevor eine Nachricht in eine Zielwarteschlange (Empfänger oder Requester) eingereiht wird.

  Dem Exit wird die gesamte Anwendungsnachricht und der gesamte Header der Übertragungswarteschlange zur Bearbeitung weitergegeben.
- Bei der Initialisierung bzw. der Beendigung des Kanals.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_SVRCONN oder MQCHT_CLNTCONN ohne Bedeutung. Für diese Kanaltypen wird niemals ein Nachrichtenexit aufgerufen.

Eine Beschreibung des Inhalts dieses Felds in verschiedenen Umgebungen finden Sie unter „MQCD - Kanaldefinition” auf Seite 1058.

Die Länge dieses Felds ist durch MQ_EXIT_NAME_LENGTH vorgegeben.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung.

MsgExitPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *MsgExit*-Felds an.

Wenn *MsgExitsDefined* größer ist als null, ist diese Adresse die Liste der Namen jedes Kanalnachrichtenexits in der Kette.

Jede Name ist ein Feld der Länge *ExitNameLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *MsgExitsDefined*-Felder, die aneinander grenzen - eins für jeden Exit.

Sämtliche Änderungen, die ein Exit an diesem Namen vornimmt, werden beibehalten, obwohl der Nachrichtenkanalexit keine explizite Aktion ausführt - welche Exits aufgerufen werden, bleibt unverändert.

Wenn *MsgExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

MsgExitsDefined (MQLONG)
Dieses Feld gibt die Anzahl der in der Kette definierten Kanalnachrichtenexits an.

Der Wert ist größer-gleich null.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

MsgRetryCount (MQLONG)
Dieses Feld gibt an, wie oft der Nachrichtenkanalagent versucht, die Nachricht einzureihen, nachdem der erste Versuch fehlgeschlagen ist.

Dieses Feld gibt an, wie oft der Nachrichtenkanalagent versucht, die Open- oder Put-Operation zu wiederholen, wenn der erste MQOPEN- oder MQPUT-Aufruf mit dem Beendigungscode MQCC_FAILED fehlschlägt.Die Wirkung dieses Attributs hängt davon ab, ob *MsgRetryExit* leer ist oder nicht:

- Wenn *MsgRetryExit* leer ist, steuert das Attribut *MsgRetryCount*, ob der Nachrichtenkanalagent versucht, die Operation zu wiederholen. Ist der Attributwert Null, werden keine Neuversuche unternommen. Ist der Attributwert größer als Null, werden Neuversuche zu den Zeitintervallen unternommen, die das Attribut *MsgRetryInterval* vorgibt.

  Neuversuche werden nur bei folgenden Ursachencodes unternommen:

  – MQRC_PAGESET_FULL
  – MQRC_PUT_INHIBITED
  – MQRC_Q_FULL

  Bei anderen Ursachencodes fährt der Nachrichtenkanalagent unverzüglich mit seiner normalen Fehlerverarbeitung fort und versucht nicht, die fehlgeschlagene Nachricht erneut einzureihen.
- Wenn *MsgRetryExit* nicht leer ist, hat das Attribut *MsgRetryCount* keine Auswirkung auf den Nachrichtenkanalagenten. Stattdessen bestimmt der Exit für Nachrichtenwiederholungen, wie oft ein Neuversuch und in welchen Zeitintervallen er unternommen wird. Der Exit wird auch dann aufgerufen, wenn das Attribut *MsgRetryCount* den Wert Null hat.

  Das Attribut *MsgRetryCount* wird dem Exit über die MQCD-Struktur zur Verfügung gestellt, muss aber nicht vom Exit beachtet werden. Neuversuche können unendlich oft fortgesetzt werden, bis der Exit im Feld MQCXP-Feld *ExitResponse* MQXCC_SUPPRESS_FUNCTION zurückgibt.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER oder MQCHT_CLUSRCVR von Bedeutung.

Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_3.

MsgRetryExit (MQCHARn)
Dieses Feld gibt den Namen des Exits für Kanalnachrichtenwiederholungen an.

Der Exit für Nachrichtenwiederholungen ist ein Exit, der vom Nachrichtenkanalagenten aufgerufen wird, wenn der Nachrichtenkanalagent den Beendigungscode MQCC_FAILED von einem MQOPEN- oder MQPUT-Aufruf erhält.Zweck dieses Exits ist es, ein Zeitintervall anzugeben, für dessen Dauer der Nachrichtenkanalagent wartet, bevor er die MQOPEN- oder MQPUT-Operation erneut ausführt. Alternativ kann der Exit so gesetzt werden, dass er nicht versucht, die Operation zu wiederholen.

Der Exit wird für alle Ursachencodes mit dem Beendigungscode MQCC_FAILED aufgerufen. Die Einstellungen des Exits legen fest, welche Ursachencodes erforderlich sind, damit der Nachrichtenkanalagent die Operation wiederholt, wie oft er sie wiederholt und in welchen Zeitintervallen.

Wenn die Operation nicht wiederholt werden soll, führt der Nachrichtenkanalagent seine normale Fehlerverarbeitung durch. Diese Verarbeitung beinhaltet das Generieren einer Abweichungsberichtsnachricht (falls vom Absender angegeben) sowie entweder das Einreihen der Originalnachricht in die Warteschlange für nicht zustellbare Nachrichten oder das Verwerfen der Nachricht (abhängig davon, ob der Absender MQRO_DEAD_LETTER_Q oder MQRO_DISCARD_MSG angegeben hat). Fehler die Warteschlange für nicht zustellbare Nachrichten betreffend (beispielsweise eine volle Warteschlange für nicht zustellbare Nachrichten) führen dazu, dass der Exit für Nachrichtenwiederholungen nicht aufgerufen wird.

Wenn der Exitname nicht leer ist, wird der Exit zu folgenden Zeiten aufgerufen:

- Unmittelbar vor dem Warten, bevor ein erneutes Zustellen der Nachricht versucht wird
- Bei Initialisierung und Beendigung des Kanals

Eine Beschreibung des Inhalts dieses Felds in verschiedenen Umgebungen finden Sie unter „MQCD - Kanaldefinition” auf Seite 1058.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER oder MQCHT_CLUSRCVR von Bedeutung.

Die Länge dieses Felds ist durch MQ_EXIT_NAME_LENGTH vorgegeben.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung.

Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_3.

MsgRetryInterval (MQLONG)
Dieses Feld gibt die Mindestzeit in Millisekunden an, nach deren Ablauf eine Open- oder Put-Operation wiederholt wird.

Die Wirkung dieses Attributs hängt davon ab, ob *MsgRetryExit* leer ist oder nicht:

- Wenn *MsgRetryExit* leer ist, gibt das Attribut *MsgRetryInterval* die Mindestzeit an, die der Nachrichtenkanalagent wartet, bevor er versucht, eine Nachricht erneut zu senden, wenn der erste MQOPEN- oder MQPUT-Aufruf mit dem Beendigungscode MQCC_FAILED fehlgeschlagen ist. Der Wert null bedeutet, dass eine Wiederholung so schnell wie möglich nach dem ersten Versuch ausgeführt wird. Neuversuche werden nur durchgeführt, wenn *MsgRetryCount* größer ist als null.

  Dieses Attribut wird auch als Wartezeit verwendet, wenn der Exit für Nachrichtenwiederholungen einen ungültigen Wert im Feld *MsgRetryInterval* in MQCXP zurückgibt.
- Wenn *MsgRetryExit* nicht leer ist, hat das Attribut *MsgRetryInterval* keine Auswirkung auf den Nachrichtenkanalagenten. Stattdessen bestimmt der Exit für Nachrichtenwiederholungen die Wartezeit des Nachrichtenkanalagenten. Das Attribut *MsgRetryInterval* wird dem Exit über die MQCD-Struktur zur Verfügung gestellt, muss aber nicht vom Exit beachtet werden.

Der Wert liegt im Bereich 0 bis 999999999.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER oder MQCHT_CLUSRCVR von Bedeutung.

Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_3.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)
Dieses Feld gibt die Benutzerdaten des Exits für Kanalnachrichtenwiederholungen an.

Diese Daten werden an den Kanalexit für Nachrichtenwiederholungen im Feld *ExitData* des Parameters *ChannelExitParms* übergeben (siehe MQ_CHANNEL_EXIT).

Dieses Feld enthält anfänglich die Daten, die in der Kanaldefinition gesetzt wurden. Allerdings werden während der Laufzeit dieser Nachrichtenkanalagent-Instanz Änderungen, die ein Exit von beliebigem Typ am Inhalt dieses Felds vorgenommen hat, vom Nachrichtenkanalagent beibehalten und den nachfolgenden Aufrufen von Exits (unabhängig vom Typ) für diese Nachrichtenkanalagent-Instanz angezeigt. Solche Änderungen haben keine Auswirkung auf die von anderen Nachrichtenkanalagent-Instanzen verwendete Kanaldefinition. Es können alle Zeichen (einschließlich Binärdaten) verwendet werden.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER oder MQCHT_CLUSRCVR von Bedeutung.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_3.

Dieses Feld ist in WebSphere MQ for IBM i ohne Bedeutung.

MsgUserData (MQCHAR32)
Dieses Feld gibt die Benutzerdaten des Kanalnachrichtenexits an.

Diese Daten werden an den Kanalnachrichtenexit im Feld *ExitData* des Parameters *ChannelExit-Parms* übergeben (siehe MQ_CHANNEL_EXIT).

Dieses Feld enthält anfänglich die Daten, die in der Kanaldefinition gesetzt wurden. Allerdings werden während der Laufzeit dieser Nachrichtenkanalagent-Instanz Änderungen, die ein Exit von beliebigem Typ am Inhalt dieses Felds vorgenommen hat, vom Nachrichtenkanalagent beibehalten und den nachfolgenden Aufrufen von Exits (unabhängig vom Typ) für diese Nachrichtenkanalagent-Instanz angezeigt. Solche Änderungen haben keine Auswirkung auf die von anderen Nachrichtenkanalagent-Instanzen verwendete Kanaldefinition. Es können alle Zeichen (einschließlich Binärdaten) verwendet werden.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

Dieses Feld ist in WebSphere MQ for IBM i ohne Bedeutung.

MsgUserDataPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *MsgUserData*-Felds an.

Wenn *MsgExitsDefined* größer ist als null, ist diese Adresse die Liste der Benutzerdatenelemente für jeden Kanalnachrichtenexit in der Kette.

Jedes Benutzerdatenelement ist ein Feld der Länge *ExitDataLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *MsgExitsDefined*-Felder, die aneinander grenzen - eins für jeden Exit. Wenn die Anzahl der definierten Benutzerdatenelemente kleiner ist als die Anzahl der Exitnamen, werden nicht definierte Benutzerdatenelemente mit Leerzeichen festgelegt. Umgekehrt gilt: Wenn die Anzahl der definierten Benutzerdatenelemente größer ist als die Anzahl der Exitnamen, werden die überzähligen Benutzerdatenelemente ignoriert und dem Exit nicht zur Verfügung gestellt.

Sämtliche Änderungen, die ein Exit an diesen Werten vornimmt, werden beibehalten. Dies ermöglicht einem Exit, einem anderen Exit Informationen zu übergeben. Da die Änderungen nicht überprüft werden, können beispielsweise Binärdaten bei Bedarf in diese Felder geschrieben werden.

Wenn *MsgExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

NetworkPriority (MQLONG)
Dieses Feld gibt die Priorität der Netzverbindung für den Kanal an.

Wenn mehrere Pfade für eine bestimmte Zieladresse verfügbar sind, wird der Pfad mit der höchsten Priorität ausgewählt. Der Wert liegt zwischen 0 bis 9, wobei 0 die niedrigste Priorität ist.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_5.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)
Dieses Feld gibt die Geschwindigkeit an, mit der nicht persistente Nachrichten durch den Kanal gesendet werden.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

Folgende Werte sind möglich:

MQNPMS_NORMAL
Normale Geschwindigkeit.

Wenn ein Kanal als MQNPMS_NORMAL definiert ist, werden nicht persistente Nachrichten in normaler Geschwindigkeit durch den Kanal gesendet. Dies hat den Vorteil, dass diese Nachrichten im Falle eines Kanalausfalls nicht verloren gehen. Zudem behalten persistente und nicht persistente Nachrichten in derselben Übertragungswarteschlange ihre relative Reihenfolge zueinander.

MQNPMS_FAST
Schnelle Geschwindigkeit.

Wenn ein Kanal als MQNPMS_FAST definiert ist, werden nicht persistente Nachrichten in schneller Geschwindigkeit durch den Kanal gesendet. Dies verbessert den Durchsatz des Kanals, bedeutet jedoch, dass nicht persistente Nachrichten im Falle eines Kanalausfalls verloren gehen. Zudem ist es möglich, dass nicht persistente Nachrichten vor persistente Nachrichten springen, die in derselben Übertragungswarteschlange warten, und damit die Reihenfolge der persistenten und nicht persistenten Nachrichten nicht beibehalten wird. Die Reihenfolge der nicht persistenten Nachrichten untereinander wird dabei jedoch nicht verändert. Analog wird auch die Reihenfolge der persistenten Nachrichten untereinander nicht verändert.

Password (MQCHAR12)
Dieses Feld gibt das Kennwort an, das vom Nachrichtenkanalagenten zur Initialisierung einer sicheren SNA-Sitzung mit einem fernen Nachrichtenkanalagenten verwendet wird.

Dieses Feld darf nur auf UNIX -Systemen und Windows nicht leer sein und ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER oder MQCHT_CLNTCONN relevant. Unter z/OS hat dieses Feld keine Bedeutung.

Die Länge dieses Felds ist durch MQ_PASSWORD_LENGTH vorgegeben. Allerdings werden nur die ersten zehn Zeichen verwendet.

Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_2ist.

PropertyControl (MQLONG)
Dieses Feld gibt an, was mit den Eigenschaften einer Nachricht geschieht, die an einen Warteschlangenmanager der Version 6 oder früher gesendet wird (diesen älteren Warteschlangenmanagern ist das Konzept Eigenschaftendeskriptor nicht bekannt).

Folgende Werte sind möglich:

MQPROP_COMPATIBILITY
Wenn die Nachricht eine Eigenschaft mit einem der folgenden Präfixe enthält, werden alle Nachrichteneigenschaften der Anwendung in einem MQRFH2-Header zugestellt: **mcd.**, **jms.**, **usr.** oder

mqext.. Andernfalls werden alle Eigenschaften der Nachricht außer denen im Nachrichtendeskriptor (oder in der Erweiterung) gelöscht und sind für die Anwendung nicht mehr zugänglich.

Dies ist der Standardwert. Er ermöglicht Anwendungen, die mit JMS-Eigenschaften in einem MQRFH2-Header in den Nachrichtendaten erwarten, unverändert fortzufahren.

MQPROP_NONE
Alle Eigenschaften der Nachricht, außer denen im Nachrichtendeskriptor (oder in der Erweiterung), werden aus der Nachricht entfernt, bevor sie an den fernen Warteschlangenmanager gesendet wird.

MQPROP_ALL
Alle Nachrichteneigenschaften sind in der Nachricht eingeschlossen, wenn sie an den fernen Warteschlangenmanager gesendet wird. Die Eigenschaften werden, mit Ausnahme der Eigenschaften im Deskriptor oder der Erweiterung der Nachricht, innerhalb der Nachrichtendaten in ein oder mehrere MQRFH2-Header eingefügt.

Dieses Attribut gilt für Sender-, Server-, Clustersender- und Clusterempfängerkanäle.

„MQIA_* (Selektoren Ganzzahlattribut)" auf Seite 115

„MQPROP_* (Warteschlangen- und Kanal-Eigenschaftensteuerungs-Werte und maximale Eigenschaftenlänge)" auf Seite 152

PutAuthority (MQLONG)
Dieses Feld gibt an, ob die Benutzer-ID in den Kontextinformationen, die mit einer Nachricht verknüpft sind, zum Erteilen der Berechtigung für das Einreihen der Nachricht in die Zielwarteschlange verwendet wird.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER oder MQCHT_CLUSRCVR von Bedeutung. Folgende Werte sind möglich:

MQPA_DEFAULT
Die standardmäßige Benutzer-ID wird verwendet.

MQPA_CONTEXT
Die Kontext-Benutzer-ID wird verwendet.

MQPA_ALTERNATE_OR_MCA
Die im Feld "UserIdentifier" des Nachrichtendeskriptors angegebene Benutzer-ID wird verwendet. Vom Netz empfangene Benutzer-IDs werden nicht übernommen. Dieser Wert wird nur unter z/OS unterstützt.

MQPA_ONLY_MCA
Die Standard-Benutzer-ID wird verwendet. Vom Netz empfangene Benutzer-IDs werden nicht übernommen. Dieser Wert wird nur unter z/OS unterstützt.

QMgrName (MQCHAR48)
Dieses Feld gibt den Namen des Warteschlangenmanagers an, zu dem der Exit eine Verbindung herstellen kann.

Bei Kanälen, deren *ChannelType* nicht MQCHT_CLNTCONN ist, enthält dieses Feld den Namen des Warteschlangenmanagers, zu dem ein Exit eine Verbindung herstellen kann. Auf UNIX-, Linux- und Windows-Systemen ist es stets leer.

Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

ReceiveExit (MQCHARn)
Dieses Feld gibt den Namen des Kanalempfangsexits an.

Wenn der Name nicht leer ist, wird der Exit zu folgenden Zeiten aufgerufen:

- Unmittelbar vor dem Verarbeiten der empfangenen Netzdaten.

  Dem Exit wird der vollständige Übertragungspuffer, wie er empfangen wurde, übergeben. Die Inhalte des Puffers können gegebenenfalls modifiziert werden.
- Bei der Initialisierung bzw. der Beendigung des Kanals.

Eine Beschreibung des Inhalts dieses Felds in verschiedenen Umgebungen finden Sie unter „MQCD - Kanaldefinition” auf Seite 1058.

Die Länge dieses Felds ist durch MQ_EXIT_NAME_LENGTH vorgegeben.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung.

ReceiveExitPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *ReceiveExit*-Felds an.

Wenn *ReceiveExitsDefined* größer ist als null, ist diese Adresse die Liste der Namen jedes Kanalempfangsexits in der Kette.

Jede Name ist ein Feld der Länge *ExitNameLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *ReceiveExitsDefined*-Felder, die aneinander grenzen - eins für jeden Exit.

Sämtliche Änderungen, die ein Exit an diesem Namen vornimmt, werden beibehalten, obwohl der Nachrichtenkanalexit keine explizite Aktion ausführt - welche Exits aufgerufen werden, bleibt unverändert.

Wenn *ReceiveExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

ReceiveExitsDefined (MQLONG)
Dieses Feld gibt die Anzahl der in der Kette definierten Kanalempfangsexits an.

Der Wert ist größer-gleich null.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

ReceiveUserData (MQCHAR32)
Dieser Kanal gibt die Benutzerdaten des Kanalempfangsexits an.

Diese Daten werden an den Kanalempfangsexit im Feld *ExitData* des Parameters *ChannelExitParms* übergeben (siehe MQ_CHANNEL_EXIT).

Dieses Feld enthält anfänglich die Daten, die in der Kanaldefinition gesetzt wurden. Allerdings werden während der Laufzeit dieser Nachrichtenkanalagent-Instanz Änderungen, die ein Exit von beliebigem Typ am Inhalt dieses Felds vorgenommen hat, vom Nachrichtenkanalagent beibehalten und den nachfolgenden Aufrufen von Exits (unabhängig vom Typ) für diese Nachrichtenkanalagent-Instanz angezeigt. Dies gilt für Exits in verschiedenen Dialogen. Solche Änderungen haben keine Auswirkung auf die von anderen Nachrichtenkanalagent-Instanzen verwendete Kanaldefinition. Es können alle Zeichen (einschließlich Binärdaten) verwendet werden.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

Dieses Feld ist in WebSphere MQ for IBM i ohne Bedeutung.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *ReceiveUserData*-Felds an.

Wenn *ReceiveExitsDefined* größer ist als null, ist diese Adresse die Liste der Benutzerdatenelemente für jeden Kanalempfangsexit in der Kette.

Jedes Benutzerdatenelement ist ein Feld der Länge *ExitDataLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *ReceiveExitsDefined*-Felder, die aneinander grenzen - eins für jeden Exit. Wenn die Anzahl der definierten Benutzerdatenelemente kleiner ist als die Anzahl der Exitnamen, werden nicht definierte Benutzerdatenelemente mit Leerzeichen festgelegt. Umgekehrt gilt: Wenn die Anzahl der

definierten Benutzerdatenelemente größer ist als die Anzahl der Exitnamen, werden die überzähligen Benutzerdatenelemente ignoriert und dem Exit nicht zur Verfügung gestellt.

Sämtliche Änderungen, die ein Exit an diesen Werten vornimmt, werden beibehalten. Dies ermöglicht einem Exit, einem anderen Exit Informationen zu übergeben. Da die Änderungen nicht überprüft werden, können beispielsweise Binärdaten bei Bedarf in diese Felder geschrieben werden.

Wenn *ReceiveExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_5.

RemotePassword (MQCHAR12)
Dieses Feld gibt das Kennwort eines Partners an.

Dieses Feld enthält nur dann gültige Informationen, wenn *ChannelType* MQCHT_CLNTCONN oder MQCHT_SVRCONN lautet.

- Bei einem Sicherheitsexit auf einem MQCHT_CLNTCONN-Kanal ist dieses Kennwort ein Kennwort, das aus der Umgebung abgerufen wurde. Der Exit kann auswählen, ob sie an den Sicherheitsexit auf dem Server gesendet wird.
- Bei einem Sicherheitsexit auf einem MQCHT_SVRCONN-Kanal kann dieses Feld ein Kennwort enthalten, das aus der Umgebung auf dem Client abgerufen wurde, wenn es keinen Clientsicherheitsexit gibt. Der Exit kann mithilfe dieses Kennworts die Benutzer-ID in *RemoteUserIdentifier* überprüfen.

  Ist ein Sicherheitsexit auf dem Client vorhanden, können diese Informationen in einem Sicherheitsfluss vom Client abgerufen werden.

Die Länge dieses Felds ist durch MQ_PASSWORD_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

RemoteSecurityId (MQBYTE40)
Dieses Feld gibt die Sicherheits-ID des Remotebenutzers an.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_CLNTCONN oder MQCHT_SVRCONN von Bedeutung.

Der folgende Sonderwert gibt an, dass keine Sicherheits-ID vorhanden ist:

MQSID_NONE
Es ist keine Sicherheits-ID angegeben.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQSID_NONE_ARRAY definiert. Diese Konstante hat den gleichen Wert wie die Konstante MQSID_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Eingabefeld für den Exit. Die Länge dieses Felds ist durch MQ_SECURITY_ID_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_7.

RemoteUserIdentifier (MQCHAR12)
Dieses Feld gibt die ersten 12 Byte der Benutzer-ID eines Partners an.

Zwei Felder enthalten die ID des Remotebenutzers:

- *RemoteUserIdentifier* enthält die ersten 12 Byte der ID des Remotebenutzers und wird mit Leerzeichen aufgefüllt, wenn die ID kürzer ist als 12 Byte. *RemoteUserIdentifier* kann leer sein.

- *LongRemoteUserIdPtr* verweist auf die vollständige ID des Remotebenutzers, die länger sein kann als 12 Byte. Ihre Länge wird durch *LongRemoteUserIdLength* vorgegeben. Die vollständige ID enthält keine abschließenden Leerzeichen und endet nicht auf null. Ist die ID leer, hat *LongRemoteUserIdLength* den Wert Null und der Wert von *LongRemoteUserIdPtr* ist nicht definiert.

  LongRemoteUserIdPtr ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_6.

Die ID des Remotebenutzers ist nur für Kanäle vom *ChannelType* MQCHT_CLNTCONN oder MQCHT_SVRCONN von Bedeutung.

- Für einen Sicherheitsexit auf einem MQCHT_CLNTCONN-Kanal ist dieser Wert eine Benutzer-ID, die aus der Umgebung abgerufen wurde. Der Exit kann auswählen, ob sie an den Sicherheitsexit auf dem Server gesendet wird.
- Für einen Sicherheitsexit auf einem MQCHT_SVRCONN-Kanal kann dieser Wert eine Benutzer-ID enthalten, die aus der Umgebung auf dem Client abgerufen wurde, wenn es keinen Clientsicherheitsexit gibt. Möglicherweise validiert der Exit diese Benutzer-ID (unter Umständen mit dem Kennwort in *RemotePassword*) und aktualisiert den Wert in *MCAUserIdentifier*.

  Ist ein Sicherheitsexit auf dem Client vorhanden, können diese Informationen in einem Sicherheitsfluss vom Client abgerufen werden.

Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_2.

SecurityExit (MQCHARn)
Dieses Feld gibt den Namen des Kanalsicherheitsexits an.

Wenn der Name nicht leer ist, wird der Exit zu folgenden Zeiten aufgerufen:

- Unmittelbar nach der Einrichtung eines Kanals.

  Bevor Nachrichten übertragen werden, erhält der Exit die Möglichkeit, Sicherheitsabläufe zu initiieren, um die Verbindungsberechtigung auszuwerten.
- Nach dem Empfang einer Antwort auf einen Sicherheitsnachrichtenfluss.

  Sämtliche Sicherheitsnachrichtenflüsse vom fernen Prozessor, die beim fernen Warteschlangenmanager eingehen, werden an den Exit weitergeleitet.
- Bei der Initialisierung bzw. der Beendigung des Kanals.

Eine Beschreibung des Inhalts dieses Feldes in verschiedenen Umgebungen finden Sie unter „MQCD - Kanaldefinition” auf Seite 1058.

Die Länge dieses Felds ist durch MQ_EXIT_NAME_LENGTH vorgegeben.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung.

SecurityUserData (MQCHAR32)
Dieser Kanal gibt die Benutzerdaten des Kanalsicherheitsexits an.

Diese Daten werden an den Kanalsicherheitsexit im Feld *ExitData* des Parameters *ChannelExitParms* übergeben (siehe MQ_CHANNEL_EXIT).

Dieses Feld enthält anfänglich die Daten, die in der Kanaldefinition gesetzt wurden. Allerdings werden während der Laufzeit dieser Nachrichtenkanalagent-Instanz Änderungen, die ein Exit von beliebigem Typ am Inhalt dieses Felds vorgenommen hat, vom Nachrichtenkanalagent beibehalten und den nachfolgenden Aufrufen von Exits (unabhängig vom Typ) für diese Nachrichtenkanalagent-Instanz angezeigt. Dies gilt für Exits in verschiedenen Dialogen. Solche Änderungen haben keine Auswirkung auf die von anderen Nachrichtenkanalagent-Instanzen verwendete Kanaldefinition. Es können alle Zeichen (einschließlich Binärdaten) verwendet werden.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

Dieses Feld ist in WebSphere MQ for IBM i ohne Bedeutung.

SendExit (MQCHARn)
Dieses Feld gibt den Namen des Kanalsendeexits an.

Wenn der Name nicht leer ist, wird der Exit zu folgenden Zeiten aufgerufen:

- Unmittelbar vor der Übertragung der Daten im Netz.

  Der Exit erhält den vollständigen Übertragungspuffer, bevor dieser übertragen wird. Die Inhalte des Puffers können gegebenenfalls modifiziert werden.
- Bei der Initialisierung bzw. der Beendigung des Kanals.

Eine Beschreibung des Inhalts dieses Felds in verschiedenen Umgebungen finden Sie unter „MQCD - Kanaldefinition” auf Seite 1058.

Die Länge dieses Felds ist durch MQ_EXIT_NAME_LENGTH vorgegeben.

Anmerkung: Der Wert dieser Konstante ist abhängig von der Umgebung.

SendExitPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *SendExit*-Felds an.

Wenn *SendExitsDefined* größer ist als null, ist diese Adresse die Liste der Namen jedes Kanalsendeexits in der Kette.

Jede Name ist ein Feld der Länge *ExitNameLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *SendExitsDefined*-Felder, die aneinander angrenzen - eins für jeden Exit.

Sämtliche Änderungen, die ein Exit an diesem Namen vornimmt, werden beibehalten, obwohl der Nachrichtensendeexit keine explizite Aktion ausführt - welche Exits aufgerufen werden, bleibt unverändert.

Wenn *SendExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

SendExitsDefined (MQLONG)
Dieses Feld gibt die Anzahl der in der Kette definierten Kanalsendeexits an.

Der Wert ist größer-gleich null.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

SendUserData (MQCHAR32)
Dieses Feld gibt die Benutzerdaten des Kanalsendeexits an.

Diese Daten werden an den Kanalsendeexit im Feld *ExitData* des Parameters *ChannelExitParms* übergeben (siehe MQ_CHANNEL_EXIT).

Dieses Feld enthält anfänglich die Daten, die in der Kanaldefinition gesetzt wurden. Allerdings werden während der Laufzeit dieser Nachrichtenkanalagent-Instanz Änderungen, die ein Exit von beliebigem Typ am Inhalt dieses Felds vorgenommen hat, vom Nachrichtenkanalagent beibehalten und den nachfolgenden Aufrufen von Exits (unabhängig vom Typ) für diese Nachrichtenkanalagent-Instanz angezeigt. Dies gilt für Exits in verschiedenen Dialogen. Solche Änderungen haben keine Auswirkung auf die von anderen Nachrichtenkanalagent-Instanzen verwendete Kanaldefinition. Es können alle Zeichen (einschließlich Binärdaten) verwendet werden.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

Dieses Feld ist in WebSphere MQ for IBM i ohne Bedeutung.

SendUserDataPtr (MQPTR)
Dieses Feld gibt die Adresse des ersten *SendUserData*-Felds an.

Wenn *SendExitsDefined* größer ist als null, ist diese Adresse die Liste der Benutzerdatenelemente für jeden Kanalnachrichtenexit in der Kette.

Jedes Benutzerdatenelement ist ein Feld der Länge *ExitDataLength*, das rechts mit Leerzeichen aufgefüllt ist. Es gibt *MsgExitsDefined*-Felder, die aneinander grenzen - eins für jeden Exit. Wenn die Anzahl der definierten Benutzerdatenelemente kleiner ist als die Anzahl der Exitnamen, werden nicht definierte Benutzerdatenelemente mit Leerzeichen festgelegt. Umgekehrt gilt: Wenn die Anzahl der definierten Benutzerdatenelemente größer ist als die Anzahl der Exitnamen, werden die überzähligen Benutzerdatenelemente ignoriert und dem Exit nicht zur Verfügung gestellt.

Sämtliche Änderungen, die ein Exit an diesen Werten vornimmt, werden beibehalten. Dies ermöglicht einem Exit, einem anderen Exit Informationen zu übergeben. Da die Änderungen nicht überprüft werden, können beispielsweise Binärdaten bei Bedarf in diese Felder geschrieben werden.

Wenn *SendExitsDefined* den Wert Null hat, ist dieses Feld der Nullzeiger.

Auf Plattformen, bei denen die Programmiersprache den Zeigerdatentyp nicht unterstützt, wird dieses Feld als Bytefolge mit entsprechender Länge deklariert.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

SeqNumberWrap (MQLONG)
Dieses Feld gibt die höchste zulässige Nachrichtenfolgenummer an.

Wenn der hier angegebene Wert erreicht ist, beginnen die Folgenummern wieder bei 1.

Dieser Wert ist nicht verhandelbar. Er muss in der lokalen und der fernen Kanaldefinition übereinstimmen.

Dieses Feld ist für Kanäle mit dem *ChannelType* MQCHT_SVRCONN oder MQCHT_CLNTCONN ohne Bedeutung.

SharingConversations (MQLONG)
Dieses Feld gibt die maximale Anzahl Dialoge an, die eine diesem Kanal zugeordnete Kanalinstanz gemeinsam nutzen können.

Dieses Feld wird für Client- und Serververbindungskanäle verwendet.

Der Wert 0 bedeutet, dass der Kanal hinsichtlich der folgenden Attribute so funktioniert wie in Versionen vor WebSphere MQ Version 7.0:

- Gemeinsame Nutzung von Dialogen
- Vorauslesen
- `STOP CHANNEL(<channelname>) MODE(QUIESCE)`
- Überwachungssignal wird gesendet
- Asynchrone Clientverarbeitung

Der Wert 1 ist der Minimalwert für das Verhalten von WebSphere MQ V7.0. Obwohl nur ein Dialog auf der Kanalinstanz zulässig ist, sind Vorauslesen, asynchrone Verarbeitung und das Verhalten der Version 7 beim `CLNTCONN-SVRCONN`-Austausch von Überwachungssignalen und Kanalstopps im Quiescemodus verfügbar.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_9 ist.

Der Standardwert für dieses Feld ist 10.

Anmerkung: Die Grenzwerte *MaxInstances* und *MaxInstancesPerClient*, die auf einen Kanal angewendet werden, beschränken die Anzahl Kanalinstanzen und nicht die Anzahl Dialoge, die diese Instanzen gemeinsam nutzen.

ShortConnectionName (MQCHAR20)
Dieses Feld gibt die ersten 20 Byte eines Verbindungsnamens an.

Wenn das Feld *Version* den Wert MQCD_VERSION_1 hat, enthält *ShortConnectionName* den vollständigen Verbindungsnamen.

Wenn das Feld *Version* den Wert MQCD_VERSION_2 oder höher hat, enthält *ShortConnectionName* die ersten 20 Zeichen des Verbindungsnamens. Der vollständige Verbindungsname wird durch das Feld *ConnectionName* angegeben. *ShortConnectionName* und die ersten 20 Zeichen von *ConnectionName* sind identisch.

Ausführliche Informationen zum Inhalt dieses Felds finden Sie unter *ConnectionName*.

Anmerkung: Der Name dieses Felds wurde für MQCD_VERSION_2 und nachfolgende Versionen von MQCD geändert. Der frühere Feldname lautete *ConnectionName*.

Die Länge dieses Felds ist durch MQ_SHORT_CONN_NAME_LENGTH vorgegeben.

ShortRetryCount (MQLONG)
Dieses Feld gibt die maximale Anzahl der Verbindungsversuche mit einem fernen System an.

Dieses Feld gibt die maximale Anzahl der Verbindungsversuche mit dem fernen System an. Die Intervalle werden vom Parameter *ShortRetryInterval* angegeben, bevor die (normalerweise längeren) Zähler *LongRetryCount* und *LongRetryInterval* zum Einsatz kommen.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

ShortRetryInterval (MQLONG)
Dieses Feld gibt die maximale Anzahl an Sekunden an, nach denen erneut versucht wird, eine Verbindung zum fernen System herzustellen.

Das Intervall zwischen zwei Verbindungsversuchen kann größer sein, wenn ein Kanal abwarten muss, bis er aktiv ist.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR oder MQCHT_CLUSRCVR von Bedeutung.

SSLCipherSpec (MQCHAR32)
Dieses Feld gibt die Verschlüsselungsspezifikation (Cipher Spec) an, die bei der Verwendung von SSL verwendet wird.

Wenn "SSLCipherSpec" leer ist, verwendet der Kanal kein SSL. Ist das Feld nicht leer, enthält es eine Zeichenfolge, die die verwendete Verschlüsselungsspezifikation angibt.

Dieser Parameter wird für alle Kanaltypen unterstützt. Es wird unter AIX, HP-UX, Linux, IBM i, Solaris, Windowsund z/OSunterstützt. Er ist nur für Kanaltypen mit dem Transporttyp (TRPTYPE) TCP zulässig.

Dies ist ein Eingabefeld für den Exit. Die Länge dieses Felds ist durch MQ_SSL_CIPHER_SPEC_LENGTH vorgegeben. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

SSLClientAuth (MQLONG)
Dieses Feld gibt an, ob eine SSL-Clientauthentifizierung erforderlich ist.

Dieses Feld ist nur für SVRCONN-Kanaldefinitionen von Bedeutung.

Folgende Werte sind möglich:

MQSCA_REQUIRED
Clientauthentifizierung erforderlich.

MQSCA_OPTIONAL
Clientauthentifizierung optional.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

SSLPeerNameLength (MQLONG)
Dieses Feld gibt die Länge (in Byte) des SSL-Peernamens an, auf den *SSLPeerNamePtr* verweist.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

SSLPeerNamePtr (MQPTR)
Dieses Feld gibt die Adresse des SSL-Peer-Namens an.

Beim Empfang eines Zertifikats während eines erfolgreichen SSL-Handshake wird der definierte Name des Zertifikatinhabers in das MQCD-Feld kopiert, auf das SSLPeerNamePtr an dem Ende des Kanals zugreift, an dem das Zertifikat empfangen wird. Dabei wird der SSLPeerName in der Kanaldefinition des lokalen Benutzers, sofern vorhanden, überschrieben. Falls an diesem Kanalende ein Sicherheitsexit definiert ist, erhält auch dieses den definierten Namen aus dem Peer-Zertifikat der MQCD.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_7 ist.

Anmerkung: Sicherheitsexitanwendungen, die vor WebSphere MQ 7.1 erstellt wurden, müssen unter Umständen aktualisiert werden. Weitere Informationen finden Sie im Abschnitt Kanalsicherheitsexitprogramme .

StrucLength (MQLONG)
Dieses Feld gibt die Länge der MQCD-Struktur in Byte an.

Die Länge schließt keine der Zeichenfolgen ein, die von Zeigerfeldern angesprochen werden, welche in der Struktur enthalten sind. Folgende Werte sind möglich:

MQCD_LENGTH_4
Länge der Kanaldefinitionsstruktur von Version 4.

MQCD_LENGTH_5
Länge der Kanaldefinitionsstruktur von Version 5.

MQCD_LENGTH_6
Länge der Kanaldefinitionsstruktur von Version 6.

MQCD_LENGTH_7
Länge der Kanaldefinitionsstruktur von Version 7.

MQCD_LENGTH_8
Länge der Kanaldefinitionsstruktur von Version 8.

MQCD_LENGTH_9
Länge der Kanaldefinitionsstruktur von Version 9.

Die folgende Konstante gibt die Länge der aktuellen Version an:

MQCD_CURRENT_LENGTH
Länge der aktuellen Version der Kanaldefinitionsstruktur.

Anmerkung: Die Werte dieser Konstanten richten sich nach der jeweils verwendeten Umgebung.

Dieses Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCD_VERSION_4.

TpName (MQCHAR64)
Dieses Feld gibt den LU 6.2-Transaktionsprogrammnamen an.

Dieses Feld ist nur von Bedeutung, wenn das Übertragungsprotokoll (*TransportType*) MQXPT_LU62 und der *ChannelType* nicht MQCHT_SVRCONN oder MQCHT_RECEIVER lautet.

Auf Plattformen, auf denen die Informationen im Kommunikationsobjekt enthalten sind, ist dieses Feld immer leer.

Die Länge dieses Felds ist durch MQ_TP_NAME_LENGTH vorgegeben.

TransportType (MQLONG)
Dieses Feld gibt das zu verwendende Übertragungsprotokoll an.

Der Wert wird nicht geprüft, wenn der Kanal von der anderen Seite initiiert wurde.

Folgende Werte sind möglich:

MQXPT_LU62
LU 6.2-Übertragungsprotokoll.

MQXPT_TCP
TCP/IP-Übertragungsprotokoll.

MQXPT_NETBIOS
NetBIOS-Übertragungsprotokoll.

Der Wert wird in den folgenden Umgebungen unterstützt: Windows.

MQXPT_SPX
SPX-Übertragungsprotokoll.

Dieser Wert wird in den folgenden Umgebungen unterstützt: Windows sowie WebSphere MQ-Clients, die mit diesen Systemen verbunden sind.

UseDLQ (MQLONG)
Dieses Feld gibt an, ob die Warteschlange für nicht zustellbare Nachrichten (oder Warteschlange für nicht zugestellte Nachrichten) verwendet wird, wenn Nachrichten nicht über Kanäle zugestellt werden können.

Es kann einen der folgenden Werte enthalten:

MQUSEDLQ_NO
Nachrichten, die von einem Kanal nicht zugestellt werden konnten, werden als Fehler behandelt. Je nach Einstellung von NPMSPEED verwirft der Kanal die Nachricht oder der Kanal wird beendet.

MQUSEDLQ_YES
Wenn das Attribut DEADQ des Warteschlangenmanagers den Namen einer Warteschlange für nicht zustellbare Nachrichten angibt, wird diese Warteschlange verwendet. Andernfalls ist das Verhalten wie bei NO. YES ist der Standardwert.

UserIdentifier (MQCHAR12)
Dieses Feld gibt die Benutzer-ID an, die vom Nachrichtenkanalagenten zur Initialisierung einer sicheren SNA-Sitzung mit einem fernen Nachrichtenkanalagenten verwendet wird.

Dieses Feld darf nur auf UNIX -Systemen und Windows nicht leer sein und ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER oder MQCHT_CLNTCONN relevant. Unter z/OS hat dieses Feld keine Bedeutung.

Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben. Allerdings werden nur die ersten zehn Zeichen verwendet.

Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCD_VERSION_2ist.

Version (MQLONG)
Das Feld `Version` gibt die höchste Versionsnummer an, die Sie für die Struktur setzen können.

Der Wert hängt von der Umgebung ab:

MQCD_VERSION_1
Kanaldefinitionsstruktur Version 1.

MQCD_VERSION_2
Kanaldefinitionsstruktur Version 2.

Version 2 wird von keinem aktuellen IBM WebSphere MQ-Produkt verwendet.

MQCD_VERSION_3
Kanaldefinitionsstruktur Version 3.

Version 3 ist die höchste Version, auf die Sie das Feld unter MQSeries Version 2 in den folgenden Umgebungen setzen können: HP Integrity NonStop Serverund UNIX and Linux -Systeme, die nicht an anderer Stelle aufgelistet sind.

MQCD_VERSION_4
Kanaldefinitionsstruktur Version 4.

Version 4 wird von keinem aktuellen IBM WebSphere MQ-Produkt verwendet.

MQCD_VERSION_5
Kanaldefinitionsstruktur Version 5.

Version 5 ist der höchste Wert, den Sie unter MQSeries for OS/390 Version 5 Release 2 in diesem Feld eingeben können.

MQCD_VERSION_6
Kanaldefinitionsstruktur Version 6.

Version 6 ist nicht die aktuelle MQCD-Strukturversion eines bestehenden IBM WebSphere MQ-Produkts. Eine MQCD-Struktur der Version 6 kann jedoch an MQCONNX unter Verwendung der Felder ClientConnOffset bzw. ClientConnPtr der MQCNO-Struktur übergeben werden.

Auf den dezentralen Plattformen ist Version 6 die Standardversion in den Initialisierern MQCD_DEFAULT und MQCD_CLIENT_CONN_DEFAULT. Wenn Sie die Felder MQCD_VERSION_7, MQCD_VERSION_8 oder MQCD_VERSION_9 des MQCD referenzieren möchten, initialisieren Sie explizit das Feld MQCD **Version** mit MQCD_VERSION_7, MQCD_VERSION_8 oder MQCD_VERSION_9 (wie erforderlich).

Unter z/OS ist MQCD_VERSION_7 der Standardwert.

MQCD_VERSION_7
Kanaldefinitionsstruktur Version 7.

Version 7 ist die höchste Version, die Sie in IBM WebSphere MQ Version 5.3 in den folgenden Umgebungen festlegen können: AIX, HP-UX, Solaris, Windowsund IBM WebSphere MQ for z/OS Version 5.3 und Version 5.3.1. MQCD_VERSION_7 ist der Standardwert für Versionen von IBM WebSphere MQ for z/OS.

MQCD_VERSION_8
Kanaldefinitionsstruktur Version 8.

Version 8 ist der höchste Wert, den Sie in IBM WebSphere MQ Version 6.0 auf sämtlichen Plattformen in diesem Feld eingeben können.

MQCD_VERSION_9
Kanaldefinitionsstruktur Version 9.

Version 9 ist der höchste Wert, den Sie in IBM WebSphere MQ Version 7.0 und IBM WebSphere MQ Version 7.0.1 auf allen Plattformen in diesem Feld eingeben können.

MQCD_VERSION_10
Kanaldefinitionsstruktur Version 10.

Version 10 ist der höchste Wert, den Sie in IBM WebSphere MQ Version 7.1 und IBM WebSphere MQ Version 7.5 auf allen Plattformen in diesem Feld eingeben können.

Felder, die nur in den jüngsten Strukturversionen existieren, sind in den Beschreibungen der Felder als solche gekennzeichnet. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCD_CURRENT_VERSION
Der in MQCD_CURRENT_VERSION gesetzte Wert ist die aktuelle Version der verwendeten Kanaldefinitionsstruktur.

Der Wert von MQCD_CURRENT_VERSION hängt von der Umgebung ab. Er enthält jeweils den höchsten von der Plattform unterstützten Wert.

MQCD_CURRENT_VERSION wird nicht zur Initialisierung der Standardstrukturen von Header, Kopie und Include-Dateien für unterschiedliche Programmiersprachen verwendet. Die Standardinitialisierung von Version hängt von der jeweiligen Plattform und vom Release ab.

Für IBM WebSphere MQ Version 7.0 und spätere Versionen werden die MQCD-Deklarationen im Header, in der Kopie und in der Include-Datei mit MQCD_VERSION_6 initialisiert. Wenn Sie weitere MQCD-Felder verwenden möchten, muss die Anwendung die Versionsnummer auf MQCD_CURRENT_VERSI-

ON setzen. Wenn Sie eine Anwendung schreiben, die zwischen mehreren Umgebungen übertragbar ist, müssen Sie eine Version verwenden, die in allen Umgebungen unterstützt wird.

Tipp: Wenn eine neue Version der MQCD-Struktur eingeführt wird, wird der Aufbau der bestehenden Komponente nicht geändert. Der Exit muss die Versionsnummer prüfen. Diese muss größer-gleich der niedrigsten Version sein, die die Felder enthält, welche der Exit verwenden muss.

XmitQName (MQCHAR48)
Dieses Feld gibt den Namen der Übertragungswarteschlange an, aus der die Nachrichten abgerufen werden.

Dieses Feld ist nur für Kanäle mit dem *ChannelType* MQCHT_SENDER oder MQCHT_SERVER von Bedeutung.

Die Länge des Felds wird durch MQ_Q_NAME_LENGTH angegeben.

Deklaration in Programmiersprache C

Bei dieser Deklaration handelt es sich um die C-Deklaration für die MQCD-Struktur.

```
typedef struct tagMQCD MQCD;
 typedef MQCD  MQPOINTER PMQCD;
 typedef PMQCD MQPOINTER PPMQCD;

 struct tagMQCD {
   MQCHAR    ChannelName[20];           /* Channel definition name */
   MQLONG    Version;                   /* Structure version number */
   MQLONG    ChannelType;               /* Channel type */
   MQLONG    TransportType;             /* Transport type */
   MQCHAR    Desc[64];                  /* Channel description */
   MQCHAR    QMgrName[48];              /* Queue-manager name */
   MQCHAR    XmitQName[48];             /* Transmission queue name */
   MQCHAR    ShortConnectionName[20];   /* First 20 bytes of */
                                        /* connection name */
   MQCHAR    MCAName[20];               /* Reserved */
   MQCHAR    ModeName[8];               /* LU 6.2 Mode name */
   MQCHAR    TpName[64];                /* LU 6.2 transaction program */
                                        /* name */
   MQLONG    BatchSize;                 /* Batch size */
   MQLONG    DiscInterval;              /* Disconnect interval */
   MQLONG    ShortRetryCount;           /* Short retry count */
   MQLONG    ShortRetryInterval;        /* Short retry wait interval */
   MQLONG    LongRetryCount;            /* Long retry count */
   MQLONG    LongRetryInterval;         /* Long retry wait interval */
   MQCHAR    SecurityExit[128];         /* Channel security exit name */
   MQCHAR    MsgExit[128];              /* Channel message exit name */
   MQCHAR    SendExit[128];             /* Channel send exit name */
   MQCHAR    ReceiveExit[128];          /* Channel receive exit name */
   MQLONG    SeqNumberWrap;             /* Highest allowable message */
                                        /* sequence number */
   MQLONG    MaxMsgLength;              /* Maximum message length */
   MQLONG    PutAuthority;              /* Put authority */
   MQLONG    DataConversion;            /* Data conversion */
   MQCHAR    SecurityUserData[32];      /* Channel security exit user */
                                        /* data */
   MQCHAR    MsgUserData[32];           /* Channel message exit user */
                                        /* data */
   MQCHAR    SendUserData[32];          /* Channel send exit user */
                                        /* data */
   MQCHAR    ReceiveUserData[32];       /* Channel receive exit user */
                                        /* data */
   /* Ver:1 */
   MQCHAR    UserIdentifier[12];        /* User identifier */
   MQCHAR    Password[12];              /* Password */
   MQCHAR    MCAUserIdentifier[12];     /* First 12 bytes of MCA user */
                                        /* identifier */
   MQLONG    MCAType;                   /* Message channel agent type */
   MQCHAR    ConnectionName[264];       /* Connection name */
   MQCHAR    RemoteUserIdentifier[12];  /* First 12 bytes of user */
                                        /* identifier from partner */
   MQCHAR    RemotePassword[12];        /* Password from partner */
   /* Ver:2 */
   MQCHAR    MsgRetryExit[128];         /* Channel message retry exit */
                                        /* name */
   MQCHAR    MsgRetryUserData[32];      /* Channel message retry exit */
                                        /* user data */
   MQLONG    MsgRetryCount;             /* Number of times MCA will */
```

```
                                    /* try to put the message, */
                                    /* after first attempt has */
                                    /* failed */
  MQLONG    MsgRetryInterval;       /* Minimum interval in */
                                    /* milliseconds after which */
                                    /* the open or put operation */
                                    /* will be retried */
  /* Ver:3 */
  MQLONG    HeartbeatInterval;      /* Time in seconds between */
                                    /* heartbeat flows */
  MQLONG    BatchInterval;          /* Batch duration */
  MQLONG    NonPersistentMsgSpeed;  /* Speed at which */
                                    /* nonpersistent messages are */
                                    /* sent */
  MQLONG    StrucLength;            /* Length of MQCD structure */
  MQLONG    ExitNameLength;         /* Length of exit name */
  MQLONG    ExitDataLength;         /* Length of exit user data */
  MQLONG    MsgExitsDefined;        /* Number of message exits */
                                    /* defined */
  MQLONG    SendExitsDefined;       /* Number of send exits */
                                    /* defined */
  MQLONG    ReceiveExitsDefined;    /* Number of receive exits */
                                    /* defined */
  MQPTR     MsgExitPtr;             /* Address of first MsgExit */
                                    /* field */
  MQPTR     MsgUserDataPtr;         /* Address of first */
                                    /* MsgUserData field */
  MQPTR     SendExitPtr;            /* Address of first SendExit */
                                    /* field */
  MQPTR     SendUserDataPtr;        /* Address of first */
                                    /* SendUserData field */
  MQPTR     ReceiveExitPtr;         /* Address of first */
                                    /* ReceiveExit field */
  MQPTR     ReceiveUserDataPtr;     /* Address of first */
                                    /* ReceiveUserData field */
  /* Ver:4 */
  MQPTR     ClusterPtr;             /* Address of a list of */
                                    /* cluster names */
  MQLONG    ClustersDefined;        /* Number of clusters to */
                                    /* which the channel belongs */
  MQLONG    NetworkPriority;        /* Network priority */
  /* Ver:5 */
  MQLONG    LongMCAUserIdLength;    /* Length of long MCA user */
                                    /* identifier */
  MQLONG    LongRemoteUserIdLength; /* Length of long remote user */
                                    /* identifier */
  MQPTR     LongMCAUserIdPtr;       /* Address of long MCA user */
                                    /* identifier */
  MQPTR     LongRemoteUserIdPtr;    /* Address of long remote */
                                    /* user identifier */
  MQBYTE40  MCASecurityId;          /* MCA security identifier */
  MQBYTE40  RemoteSecurityId;       /* Remote security identifier */
  /* Ver:6 */
  MQCHAR    SSLCipherSpec[32];      /* SSL CipherSpec */
  MQPTR     SSLPeerNamePtr;         /* Address of SSL peer name */
  MQLONG    SSLPeerNameLength;      /* Length of SSL peer name */
  MQLONG    SSLClientAuth;          /* Whether SSL client */
                                    /* authentication is required */
  MQLONG    KeepAliveInterval;      /* Keepalive interval */
  MQCHAR    LocalAddress[48];       /* Local communications */
                                    /* address */
  MQLONG    BatchHeartbeat;         /* Batch heartbeat interval */
  /* Ver:7 */
  MQLONG    HdrCompList[2];         /* Header data compression */
                                    /* list */
  MQLONG    MsgCompList[16];        /* Message data compression */
                                    /* list */
  MQLONG    CLWLChannelRank;        /* Channel rank */
  MQLONG    CLWLChannelPriority;    /* Channel priority */
  MQLONG    CLWLChannelWeight;      /* Channel weight */
  MQLONG    ChannelMonitoring;      /* Channel monitoring */
  MQLONG    ChannelStatistics;      /* Channel statistics */
  /* Ver:8 */
  MQLONG    SharingConversations;   /* Limit on sharing */
                                    /* conversations */
  MQLONG    PropertyControl;        /* Message property control */
  MQLONG    MaxInstances;           /* Limit on SVRCONN channel */
                                    /* instances */
  MQLONG    MaxInstancesPerClient;  /* Limit on SVRCONN channel */
                                    /* instances per client */
  MQLONG    ClientChannelWeight;    /* Client channel weight */
  MQLONG    ConnectionAffinity;     /* Connection affinity */
```

```
   /* Ver:9 */
   MQLONG    BatchDataLimit;           /* Batch data limit */
   MQLONG    UseDLQ;                   /* Use Dead Letter Queue */
   MQLONG    DefReconnect;             /* Default client reconnect */
                                       /* option */
   /* Ver:10 */
 };
```

COBOL-Delaration

Bei dieser Deklaration handelt es sich um die COBOL-Deklaration für die MQCD-Struktur.

```
** MQCD structure
       10  MQCD.
      ** Channel definition name
       15  MQCD-CHANNELNAME PIC X(20).
      ** Structure version number
       15  MQCD-VERSION PIC S9(9) BINARY.
      ** Channel type
       15  MQCD-CHANNELTYPE PIC S9(9) BINARY.
      ** Transport type
       15  MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
      ** Channel description
       15  MQCD-DESC PIC X(64).
      ** Queue-manager name
       15  MQCD-QMGRNAME PIC X(48).
      ** Transmission queue name
       15  MQCD-XMITQNAME PIC X(48).
      ** First 20 bytes of connection name
       15  MQCD-SHORTCONNECTIONNAME PIC X(20).
      ** Reserved
       15  MQCD-MCANAME PIC X(20).
      ** LU 6.2 Mode name
       15  MQCD-MODENAME PIC X(8).
      ** LU 6.2 transaction program name
       15  MQCD-TPNAME PIC X(64).
      ** Batch size
       15  MQCD-BATCHSIZE PIC S9(9) BINARY.
      ** Disconnect interval
       15  MQCD-DISCINTERVAL PIC S9(9) BINARY.
      ** Short retry count
       15  MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
      ** Short retry wait interval
       15  MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
      ** Long retry count
       15  MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
      ** Long retry wait interval
       15  MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
      ** Channel security exit name
       15  MQCD-SECURITYEXIT PIC X(20).
      ** Channel message exit name
       15  MQCD-MSGEXIT PIC X(20).
      ** Channel send exit name
       15  MQCD-SENDEXIT PIC X(20).
      ** Channel receive exit name
       15  MQCD-RECEIVEEXIT PIC X(20).
      ** Highest allowable message sequence number
       15  MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
      ** Maximum message length
       15  MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
      ** Put authority
       15  MQCD-PUTAUTHORITY PIC S9(9) BINARY.
      ** Data conversion
       15  MQCD-DATACONVERSION PIC S9(9) BINARY.
      ** Channel security exit user data
       15  MQCD-SECURITYUSERDATA PIC X(32).
      ** Channel message exit user data
       15  MQCD-MSGUSERDATA PIC X(32).
      ** Channel send exit user data
       15  MQCD-SENDUSERDATA PIC X(32).
      ** Channel receive exit user data
       15  MQCD-RECEIVEUSERDATA PIC X(32).
      ** Ver:1 **
      ** User identifier
       15  MQCD-USERIDENTIFIER PIC X(12).
      ** Password
       15  MQCD-PASSWORD PIC X(12).
      ** First 12 bytes of MCA user identifier
       15  MQCD-MCAUSERIDENTIFIER PIC X(12).
      ** Message channel agent type
```

```
        15  MQCD-MCATYPE PIC S9(9) BINARY.
       ** Connection name
        15  MQCD-CONNECTIONNAME PIC X(264).
       ** First 12 bytes of user identifier from partner
        15  MQCD-REMOTEUSERIDENTIFIER PIC X(12).
       ** Password from partner
        15  MQCD-REMOTEPASSWORD PIC X(12).
       ** Ver:2 **
       ** Channel message retry exit name
        15  MQCD-MSGRETRYEXIT PIC X(20).
       ** Channel message retry exit user data
        15  MQCD-MSGRETRYUSERDATA PIC X(32).
       ** Number of times MCA will try to put the message, after first
       ** attempt has failed
        15  MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
       ** Minimum interval in milliseconds after which the open or put
       ** operation will be retried
        15  MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
       ** Ver:3 **
       ** Time in seconds between heartbeat flows
        15  MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
       ** Batch duration
        15  MQCD-BATCHINTERVAL PIC S9(9) BINARY.
       ** Speed at which nonpersistent messages are sent
        15  MQCD-NONPERSISTENTMSGSPEED PIC S9(9) BINARY.
       ** Length of MQCD structure
        15  MQCD-STRUCLENGTH PIC S9(9) BINARY.
       ** Length of exit name
        15  MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
       ** Length of exit user data
        15  MQCD-EXITDATALENGTH PIC S9(9) BINARY.
       ** Number of message exits defined
        15  MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
       ** Number of send exits defined
        15  MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
       ** Number of receive exits defined
        15  MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
       ** Address of first MsgExit field
        15  MQCD-MSGEXITPTR POINTER.
       ** Address of first MsgUserData field
        15  MQCD-MSGUSERDATAPTR POINTER.
       ** Address of first SendExit field
        15  MQCD-SENDEXITPTR POINTER.
       ** Address of first SendUserData field
        15  MQCD-SENDUSERDATAPTR POINTER.
       ** Address of first ReceiveExit field
        15  MQCD-RECEIVEEXITPTR POINTER.
       ** Address of first ReceiveUserData field
        15  MQCD-RECEIVEUSERDATAPTR POINTER.
       ** Ver:4 **
       ** Address of a list of cluster names
        15  MQCD-CLUSTERPTR POINTER.
       ** Number of clusters to which the channel belongs
        15  MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
       ** Network priority
        15  MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
       ** Ver:5 **
       ** Length of long MCA user identifier
        15  MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
       ** Length of long remote user identifier
        15  MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
       ** Address of long MCA user identifier
        15  MQCD-LONGMCAUSERIDPTR POINTER.
       ** Address of long remote user identifier
        15  MQCD-LONGREMOTEUSERIDPTR POINTER.
       ** MCA security identifier
        15  MQCD-MCASECURITYID PIC X(40).
       ** Remote security identifier
        15  MQCD-REMOTESECURITYID PIC X(40).
       ** Ver:6 **
       ** SSL CipherSpec
        15  MQCD-SSLCIPHERSPEC PIC X(32).
       ** Address of SSL peer name
        15  MQCD-SSLPEERNAMEPTR POINTER.
       ** Length of SSL peer name
        15  MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
       ** Whether SSL client authentication is required
        15  MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
       ** Keepalive interval
        15  MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
       ** Local communications address
        15  MQCD-LOCALADDRESS PIC X(48).
```

```
      ** Batch heartbeat interval
       15  MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
      ** Ver:7 **
      ** Header data compression list
       15  MQCD-HDRCOMPLIST PIC S9(9) BINARY.
      ** Message data compression list
       15  MQCD-MSGCOMPLIST PIC S9(9) BINARY.
      ** Channel rank
       15  MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
      ** Channel priority
       15  MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
      ** Channel weight
       15  MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
      ** Channel monitoring
       15  MQCD-CHANNELMONITORING PIC S9(9) BINARY.
      ** Channel statistics
       15  MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
      ** Ver:8 **
      ** Limit on sharing conversations
       15  MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
      ** Message property control
       15  MQCD-PROPERTYCONTROL PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances
       15  MQCD-MAXINSTANCES PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances per client
       15  MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
      ** Client channel weight
       15  MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
      ** Connection affinity
       15  MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
      ** Ver:9 **
      ** Batch data limit
       15  MQCD-BATCHDATALIMIT PIC S9(9) BINARY.
      ** Use Dead Letter Queue
       15  MQCD-USEDLQ PIC S9(9) BINARY.
      ** Default client reconnect option
       15  MQCD-DEFRECONNECT PIC S9(9) BINARY.
      ** Ver:10 **
```

Deklaration in RPG (ILE)

Bei dieser Deklaration handelt es sich um die RPG-Deklaration für die MQCD-Struktur.

```
D* MQCD Structure
     D*
     D* Channel definition name
     D  CDCHN                  1     20
     D* Structure version number
     D  CDVER                 21     24I 0
     D* Channel type
     D  CDCHT                 25     28I 0
     D* Transport type
     D  CDTRT                 29     32I 0
     D* Channel description
     D  CDDES                 33     96
     D* Queue-manager name
     D  CDQM                  97    144
     D* Transmission queue name
     D  CDXQ                 145    192
     D* First 20 bytes of connection name
     D  CDSCN                193    212
     D* Reserved
     D  CDMCA                213    232
     D* LU 6.2 Mode name
     D  CDMOD                233    240
     D* LU 6.2 transaction program name
     D  CDTP                 241    304
     D* Batch size
     D  CDBS                 305    308I 0
     D* Disconnect interval
     D  CDDI                 309    312I 0
     D* Short retry count
     D  CDSRC                313    316I 0
     D* Short retry wait interval
     D  CDSRI                317    320I 0
     D* Long retry count
     D  CDLRC                321    324I 0
     D* Long retry wait interval
     D  CDLRI                325    328I 0
     D* Channel security exit name
```

```
     D  CDSCX                329    348
     D* Channel message exit name
     D  CDMSX                349    368
     D* Channel send exit name
     D  CDSNX                369    388
     D* Channel receive exit name
     D  CDRCX                389    408
     D* Highest allowable message sequence number
     D  CDSNW                409    412I 0
     D* Maximum message length
     D  CDMML                413    416I 0
     D* Put authority
     D  CDPA                 417    420I 0
     D* Data conversion
     D  CDDC                 421    424I 0
     D* Channel security exit user data
     D  CDSCD                425    456
     D* Channel message exit user data
     D  CDMSD                457    488
     D* Channel send exit user data
     D  CDSND                489    520
     D* Channel receive exit user data
     D  CDRCD                521    552
     D* Ver:1 **
     D* User identifier
     D  CDUID                553    564
     D* Password
     D  CDPW                 565    576
     D* First 12 bytes of MCA user identifier
     D  CDAUI                577    588
     D* Message channel agent type
     D  CDCAT                589    592I 0
     D* Connection name
     D  CDCON                593    848
     D  CDCN2                849    856
     D* First 12 bytes of user identifier from partner
     D  CDRUI                857    868
     D* Password from partner
     D  CDRPW                869    880
     D* Ver:2 **
     D* Channel message retry exit name
     D  CDMRX                881    900
     D* Channel message retry exit user data
     D  CDMRD                901    932
     D* Number of times MCA will try to put the message, after first
     D* attempt has failed
     D  CDMRC                933    936I 0
     D* Minimum interval in milliseconds after which the open or put
     D* operation will be retried
     D  CDMRI                937    940I 0
     D* Ver:3 **
     D* Time in seconds between heartbeat flows
     D  CDHBI                941    944I 0
     D* Batch duration
     D  CDBI                 945    948I 0
     D* Speed at which nonpersistent messages are sent
     D  CDNPM                949    952I 0
     D* Length of MQCD structure
     D  CDLEN                953    956I 0
     D* Length of exit name
     D  CDXNL                957    960I 0
     D* Length of exit user data
     D  CDXDL                961    964I 0
     D* Number of message exits defined
     D  CDMXD                965    968I 0
     D* Number of send exits defined
     D  CDSXD                969    972I 0
     D* Number of receive exits defined
     D  CDRXD                973    976I 0
     D* Address of first MsgExit field
     D  CDMXP                977    992*
     D* Address of first MsgUserData field
     D  CDMUP                993   1008*
     D* Address of first SendExit field
     D  CDSXP               1009   1024*
     D* Address of first SendUserData field
     D  CDSUP               1025   1040*
     D* Address of first ReceiveExit field
     D  CDRXP               1041   1056*
     D* Address of first ReceiveUserData field
     D  CDRUP               1057   1072*
     D* Ver:4 **
```

```
     D* Address of a list of cluster names
     D  CDCLP               1073   1088*
     D* Number of clusters to which the channel belongs
     D  CDCLD               1089   1092I 0
     D* Network priority
     D  CDNP                1093   1096I 0
     D* Ver:5 **
     D* Length of long MCA user identifier
     D  CDLML               1097   1100I 0
     D* Length of long remote user identifier
     D  CDLRL               1101   1104I 0
     D* Address of long MCA user identifier
     D  CDLMP               1105   1120*
     D* Address of long remote user identifier
     D  CDLRP               1121   1136*
     D* MCA security identifier
     D  CDMSI               1137   1176
     D* Remote security identifier
     D  CDRSI               1177   1216
     D* Ver:6 **
     D* SSL CipherSpec
     D  CDSCS               1217   1248
     D* Address of SSL peer name
     D  CDSPN               1249   1264*
     D* Length of SSL peer name
     D  CDSPL               1265   1268I 0
     D* Whether SSL client authentication is required
     D  CDSCA               1269   1272I 0
     D* Keepalive interval
     D  CDKAI               1273   1276I 0
     D* Local communications address
     D  CDLOA               1277   1324
     D* Batch heartbeat interval
     D  CDBHB               1325   1328I 0
     D* Ver:7 **
     D* Header data compression list
     D  CDHCL0
     D  CDHCL1              1329   1332I 0
     D  CDHCL2              1333   1336I 0
     D  CDHCL                        10I 0 DIM(2) OVERLAY(CDHCL0)
     D* Message data compression list
     D  CDMCL0
     D  CDMCL1              1337   1340I 0
     D  CDMCL2              1341   1344I 0
     D  CDMCL3              1345   1348I 0
     D  CDMCL4              1349   1352I 0
     D  CDMCL5              1353   1356I 0
     D  CDMCL6              1357   1360I 0
     D  CDMCL7              1361   1364I 0
     D  CDMCL8              1365   1368I 0
     D  CDMCL9              1369   1372I 0
     D  CDMCL10             1373   1376I 0
     D  CDMCL11             1377   1380I 0
     D  CDMCL12             1381   1384I 0
     D  CDMCL13             1385   1388I 0
     D  CDMCL14             1389   1392I 0
     D  CDMCL15             1393   1396I 0
     D  CDMCL16             1397   1400I 0
     D  CDMCL                        10I 0 DIM(16) OVERLAY(CDMCL0)
     D* Channel rank
     D  CDCWCR              1401   1404I 0
     D* Channel priority
     D  CDCWCP              1405   1408I 0
     D* Channel weight
     D  CDCWCW              1409   1412I 0
     D* Channel monitoring
     D  CDCHLMON            1413   1416I 0
     D* Channel statistics
     D  CDCHLST             1417   1420I 0
     D* Ver:8 **
     D* Limit on sharing conversations
     D  CDSHC               1421   1424I 0
     D* Message property control
     D  CDPRC               1425   1428I 0
     D* Limit on SVRCONN channel instances
     D  CDMXIN              1429   1432I 0
     D* Limit on SVRCONN channel instances per client
     D  CDMXIC              1433   1436I 0
     D* Client channel weight
     D  CDCLNCHLW           1437   1440I 0
     D* Connection affinity
     D  CDCONNAFF           1441   1444I 0
```

```
     D* Ver:9 **
     D* Batch data limit
     D  CDBDL              1445   1448I 0
     D* Use Dead Letter Queue
     D  CDUDLQ             1449   1452I 0
     D* Default client reconnect option
     D  CDDRCN             1453   1456I 0
     D* Ver:10 **
```

Deklaration in System/390 Assembler

Bei dieser Deklaration handelt es sich um die System/390-Assemblerdeklaration für die MQCD-Struktur.

```
MQCD                          DSECT
MQCD_CHANNELNAME              DS   CL20   Channel definition name
MQCD_VERSION                  DS   F      Structure version number
MQCD_CHANNELTYPE              DS   F      Channel type
MQCD_TRANSPORTTYPE            DS   F      Transport type
MQCD_DESC                     DS   CL64   Channel description
MQCD_QMGRNAME                 DS   CL48   Queue-manager name
MQCD_XMITQNAME                DS   CL48   Transmission queue name
MQCD_SHORTCONNECTIONNAME      DS   CL20   First 20 bytes of connection
*                                         name
MQCD_MCANAME                  DS   CL20   Reserved
MQCD_MODENAME                 DS   CL8    LU 6.2 Mode name
MQCD_TPNAME                   DS   CL64   LU 6.2 transaction program name
MQCD_BATCHSIZE                DS   F      Batch size
MQCD_DISCINTERVAL             DS   F      Disconnect interval
MQCD_SHORTRETRYCOUNT          DS   F      Short retry count
MQCD_SHORTRETRYINTERVAL       DS   F      Short retry wait interval
MQCD_LONGRETRYCOUNT           DS   F      Long retry count
MQCD_LONGRETRYINTERVAL        DS   F      Long retry wait interval
MQCD_SECURITYEXIT             DS   CLn    Channel security exit name
MQCD_MSGEXIT                  DS   CLn    Channel message exit name
MQCD_SENDEXIT                 DS   CLn    Channel send exit name
MQCD_RECEIVEEXIT              DS   CLn    Channel receive exit name
MQCD_SEQNUMBERWRAP            DS   F      Highest allowable message
*                                         sequence number
MQCD_MAXMSGLENGTH             DS   F      Maximum message length
MQCD_PUTAUTHORITY             DS   F      Put authority
MQCD_DATACONVERSION           DS   F      Data conversion
MQCD_SECURITYUSERDATA         DS   CL32   Channel security exit user data
MQCD_MSGUSERDATA              DS   CL32   Channel message exit user data
MQCD_SENDUSERDATA             DS   CL32   Channel send exit user data
MQCD_RECEIVEUSERDATA          DS   CL32   Channel receive exit user data
MQCD_USERIDENTIFIER           DS   CL12   User identifier
MQCD_PASSWORD                 DS   CL12   Password
MQCD_MCAUSERIDENTIFIER        DS   CL12   First 12 bytes of MCA user
*                                         identifier
MQCD_MCATYPE                  DS   F      Message channel agent type
MQCD_CONNECTIONNAME           DS   CL264  Connection name
MQCD_REMOTEUSERIDENTIFIER     DS   CL12   First 12 bytes of user
*                                         identifier from partner
MQCD_REMOTEPASSWORD           DS   CL12   Password from partner
MQCD_MSGRETRYEXIT             DS   CLn    Channel message retry exit name
MQCD_MSGRETRYUSERDATA         DS   CL32   Channel message retry exit user
*                                         data
MQCD_MSGRETRYCOUNT            DS   F      Number of times MCA will try to
*                                         put the message, after the
*                                         first attempt has failed
MQCD_MSGRETRYINTERVAL         DS   F      Minimum interval in
*                                         milliseconds after which the
*                                         open or put operation will be
*                                         retried
MQCD_HEARTBEATINTERVAL        DS   F      Time in seconds between
*                                         heartbeat flows
MQCD_BATCHINTERVAL            DS   F      Batch duration
MQCD_NONPERSISTENTMSGSPEED    DS   F      Speed at which nonpersistent
*                                         messages are sent
MQCD_STRUCLENGTH              DS   F      Length of MQCD structure
MQCD_EXITNAMELENGTH           DS   F      Length of exit name
MQCD_EXITDATALENGTH           DS   F      Length of exit user data
MQCD_MSGEXITSDEFINED          DS   F      Number of message exits defined
MQCD_SENDEXITSDEFINED         DS   F      Number of send exits defined
MQCD_RECEIVEEXITSDEFINED      DS   F      Number of receive exits defined
MQCD_MSGEXITPTR               DS   F      Address of first MSGEXIT field
MQCD_MSGUSERDATAPTR           DS   F      Address of first MSGUSERDATA
*                                         field
MQCD_SENDEXITPTR              DS   F      Address of first SENDEXIT field
```

```
MQCD_SENDUSERDATAPTR         DS   F      Address of first SENDUSERDATA
*                                        field
MQCD_RECEIVEEXITPTR          DS   F      Address of first RECEIVEEXIT
*                                        field
MQCD_RECEIVEUSERDATAPTR      DS   F      Address of first
*                                        RECEIVEUSERDATA field
MQCD_CLUSTERPTR              DS   F      Address of a list of cluster
*                                        names
MQCD_CLUSTERSDEFINED         DS   F      Number of clusters to which the
*                                        channel belongs
MQCD_NETWORKPRIORITY         DS   F      Network priority
MQCD_LONGMCAUSERIDLENGTH     DS   F      Length of long MCA user
*                                        identifier
MQCD_LONGREMOTEUSERIDLENGTH  DS   F      Length of long remote user
*                                        identifier
MQCD_LONGMCAUSERIDPTR        DS   F      Address of long MCA user
*                                        identifier
MQCD_LONGREMOTEUSERIDPTR     DS   F      Address of long remote user
*                                        identifier
MQCD_MCASECURITYID           DS   XL40   MCA security identifier
MQCD_REMOTESECURITYID        DS   XL40   Remote security identifier
MQCD_SSLCIPHERSPEC           DS   CL32   SSL CipherSpec
MQCD_SSLPEERNAMEPTR          DS   F      Address of SSL peer name
MQCD_SSLPEERNAMELENGTH       DS   F      Length of SSL peer name
MQCD_SSLCLIENTAUTH           DS   F      Whether SSL client
*                                        authentication is required
MQCD_KEEPALIVEINTERVAL       DS   F      Keepalive interval
MQCD_LOCALADDRESS            DS   CL48   Local communications address
MQCD_BATCHHEARTBEAT          DS   F      Batch heartbeat interval
MQCD_HDRCOMPLIST             DS   CL2    Header data compression list
MQCD_MSGCOMPLIST             DS   CL16   Message data compression list
MQCD_CLWLCHANNELRANK         DS   F      Channel rank
MQCD_CLWLCHANNELPRIORITY     DS   F      Channel priority
MQCD_CLWLCHANNELWEIGHT       DS   F      Channel weight
MQCD_CHANNELMONITORING       DS   F      Channel monitoring
MQCD_CHANNELSTATISTICS       DS   F      Channel statistics
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing
*                                        conversations
MQCD_PROPERTYCONTROL         DS   F      Message property
*                                        control
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing conversations
MQCD_PROPERTYCONTROL         DS   F      Message property control
MQCD_MAXINSTANCES            DS   F      Limit on SVRCONN chl instances
MQCD_MAXINSTANCESPERCLIENT   DS   F      Limit on SVRCONN chl instances
                                         per client
MQCD_CLIENTCHANNELWEIGHT     DS   F      Channel weight
MQCD_CONNECTIONAFFINITY      DS   F      Connection Affinty
MQCD_BATCHDATALIMIT          DS   F      Batch data limit
MQCD_USEDLQ                  DS   F      Use dead-letter queue
MQCD_DEFRECONNECT            DS   F      Default client reconnect option
MQCD_LENGTH                  EQU  *-MQCD
                             ORG  MQCD
MQCD_AREA                    DS   CL(MQCD_LENGTH)
```

Deklaration in Visual Basic

Bei dieser Deklaration handelt es sich um die Visual Basic-Deklaration der MQCD-Struktur.

In Visual Basic kann die MQCD-Struktur zusammen mit der MQCNO-Struktur im MQCONNX-Aufruf verwendet werden.

```
Type MQCD
  ChannelName            As String*20  'Channel definition name'
  Version                As Long       'Structure version number'
  ChannelType            As Long       'Channel type'
  TransportType          As Long       'Transport type'
  Desc                   As String*64  'Channel description'
  QMgrName               As String*48  'Queue-manager name'
  XmitQName              As String*48  'Transmission queue name'
  ShortConnectionName    As String*20  'First 20 bytes of connection'
                                       'name'
  MCAName                As String*20  'Reserved'
  ModeName               As String*8   'LU 6.2 Mode name'
  TpName                 As String*64  'LU 6.2 transaction program name'
  BatchSize              As Long       'Batch size'
  DiscInterval           As Long       'Disconnect interval'
  ShortRetryCount        As Long       'Short retry count'
  ShortRetryInterval     As Long       'Short retry wait interval'
  LongRetryCount         As Long       'Long retry count'
```

```
  LongRetryInterval      As Long       'Long retry wait interval'
  SecurityExit           As String*128 'Channel security exit name'
  MsgExit                As String*128 'Channel message exit name'
  SendExit               As String*128 'Channel send exit name'
  ReceiveExit            As String*128 'Channel receive exit name'
  SeqNumberWrap          As Long       'Highest allowable message'
                                       'sequence number'
  MaxMsgLength           As Long       'Maximum message length'
  PutAuthority           As Long       'Put authority'
  DataConversion         As Long       'Data conversion'
  SecurityUserData       As String*32  'Channel security exit user data'
  MsgUserData            As String*32  'Channel message exit user data'
  SendUserData           As String*32  'Channel send exit user data'
  ReceiveUserData        As String*32  'Channel receive exit user data'
  UserIdentifier         As String*12  'User identifier'
  Password               As String*12  'Password'
  MCAUserIdentifier      As String*12  'First 12 bytes of MCA user'
                                       'identifier'
  MCAType                As Long       'Message channel agent type'
  ConnectionName         As String*264 'Connection name'
  RemoteUserIdentifier   As String*12  'First 12 bytes of user'
                                       'identifier from partner'
  RemotePassword         As String*12  'Password from partner'
  MsgRetryExit           As String*128 'Channel message retry exit name'
  MsgRetryUserData       As String*32  'Channel message retry exit user'
                                       'data'
  MsgRetryCount          As Long       'Number of times MCA will try to'
                                       'put the message, after the'
                                       'first attempt has failed'
  MsgRetryInterval       As Long       'Minimum interval in'
                                       'milliseconds after which the'
                                       'open or put operation will be'
                                       'retried'
  HeartbeatInterval      As Long       'Time in seconds between'
                                       'heartbeat flows'
  BatchInterval          As Long       'Batch duration'
  NonPersistentMsgSpeed  As Long       'Speed at which nonpersistent'
                                       'messages are sent'
  StrucLength            As Long       'Length of MQCD structure'
  ExitNameLength         As Long       'Length of exit name'
  ExitDataLength         As Long       'Length of exit user data'
  MsgExitsDefined        As Long       'Number of message exits defined'
  SendExitsDefined       As Long       'Number of send exits defined'
  ReceiveExitsDefined    As Long       'Number of receive exits defined'
  MsgExitPtr             As MQPTR      'Address of first MsgExit field'
  MsgUserDataPtr         As MQPTR      'Address of first MsgUserData'
                                       'field'
  SendExitPtr            As MQPTR      'Address of first SendExit field'
  SendUserDataPtr        As MQPTR      'Address of first SendUserData'
                                       'field'
  ReceiveExitPtr         As MQPTR      'Address of first ReceiveExit'
                                       'field'
  ReceiveUserDataPtr     As MQPTR      'Address of first'
                                       'ReceiveUserData field'
  ClusterPtr             As MQPTR      'Address of a list of cluster'
                                       'names'
  ClustersDefined        As Long       'Number of clusters to which the'
                                       'channel belongs'
  NetworkPriority        As Long       'Network priority'
  LongMCAUserIdLength    As Long       'Length of long MCA user'
                                       'identifier'
  LongRemoteUserIdLength As Long       'Length of long remote user'
                                       'identifier'
  LongMCAUserIdPtr       As MQPTR      'Address of long MCA user'
                                       'identifier'
  LongRemoteUserIdPtr    As MQPTR      'Address of long remote user'
                                       'identifier'
  MCASecurityId          As MQBYTE40   'MCA security identifier'
  RemoteSecurityId       As MQBYTE40   'Remote security identifier'
  SSLCipherSpec          As String*32  'SSL CipherSpec'
  SSLPeerNamePtr         As MQPTR      'Address of SSL peer name'
  SSLPeerNameLength      As Long       'Length of SSL peer name'
  SSLClientAuth          As Long       'Whether SSL client'
                                       'authentication is required'
  KeepAliveInterval      As Long       'Keepalive interval'
  LocalAddress           As String*48  'Local communications address'
  BatchHeartbeat         As Long       'Batch heartbeat interval'
  HdrCompList(0 to 1)    As Long2      'Header data compression list'
  MsgCompList(0 To 15)   As Long16     'Message data compression list'
  CLWLChannelRank        As Long       'Channel Rank'
  CLWLChannelPriority    As Long       'Channel priority'
  CLWLChannelWeight      As Long       'Channel Weight'
```

```
  ChannelMonitoring      As Long       'Channel Monitoring control'
  ChannelStatistics      As Long       'Channel Statistics'
End Type
```

Ändern von MQCD-Feldern in einem Kanalexit

Die Felder im MQCD können vom Kanalexit geändert werden. Diese Änderungen werden jedoch nur unter den nachfolgend aufgeführten Bedingungen berücksichtigt.

Wenn ein Kanalexitprogramm ein Feld in der MQCD-Struktur ändert, wird der neue Wert in der Regel vom WebSphere MQ-Kanalprozess ignoriert. Der neue Wert verbleibt jedoch im MQCD und wird an alle verbleibenden Exits in einer Exitkette übergeben sowie an jeden Datenaustausch mit einer gemeinsamen Nutzung der Kanalinstanz.

Wenn SharingConversations in der MQCXP-Struktur auf FALSE eingestellt ist, werden Änderungen bestimmter Felder entsprechend dem Typ des Exitprogramms, dem Kanaltyp und dem Ursachencode des Exits berücksichtigt. In der folgenden Tabelle sind die Felder aufgeführt, die geändert werden können und sich auf das Verhalten des Kanals auswirken, und die jeweiligen Bedingungen. Wenn ein Exitprogramm eines dieser Felder bei anderen Bedingungen oder ein nicht aufgeführtes Feld ändert, wird der neue Wert durch den Kanalprozess geändert. Der neue Wert verbleibt im MQCD und wird an alle verbleibenden Exits einer Exitkette sowie an alle Dialoge übergeben, die die Kanalinstanz gemeinsam nutzen.

Jedes beliebige Exitprogramm, das zur Initialisierung aufgerufen wird (MQXR_INIT), kann das Feld ChannelName für jeden Kanaltyp ändern, wenn SharingConverstions in der MQCXP-Struktur auf FALSE eingestellt ist. Das Feld MCAUserIdentifier kann unabhängig von SharingConversations in der MQCXP-Struktur nur durch einen Sicherheitsexit geändert werden.

Feld	**Ursachencode des Exits**	**Exittyp**	**Kanaltyp**
ChannelName	MQXR_INIT	Alle	Alle
TransportType	MQXR_INIT	Alle	Alle
XmitQName	MQXR_INIT	Alle	SDR, RCVR
ModeName	MQXR_INIT	Alle	Alle
TpName	MQXR_INIT	Alle	Alle
BatchSize	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
ShortRetryCount	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
ShortRetryInterval	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR

Feld	Ursachencode des Exits	Exittyp	Kanaltyp
LongRetryCount	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
LongRetryInterval	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SeqNumberWrap	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgLength	MQXR_INIT	Alle	Alle
PutAuthority	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Alle	Alle
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Sicherheit	RCVR, RQSTR, SVRCONN, CLUSRCVR
ConnectionName	MQXR_INIT	Alle	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	MQXR_INIT	Alle	RCVR, RQSTR, CLUSRCVR
MsgRetryCount	MQXR_INIT	Alle	RCVR, RQSTR, CLUSRCVR
MsgRetryInterval	MQXR_INIT	Alle	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Alle	Alle
BatchInterval	MQXR_INIT	Alle	SDR, SVR, CLUSSDR, CLUSRCVR

Feld	Ursachencode des Exits	Exittyp	Kanaltyp
NonPersistentMsgSpeed	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Sicherheit	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Alle	Alle
SSLPeerNamePtr	MQXR_INIT	Alle	Alle
SSLPeerNameLength	MQXR_INIT	Alle	Alle
SSLClientAuth	MQXR_INIT	Alle	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
KeepAliveInterval	MQXR_INIT	Alle	Alle
LocalAddress	MQXR_INIT	Alle	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	MQXR_INIT	Alle	SDR, SVR, CLUSSDR, CLUSRCVR
HdrCompList	MQXR_INIT	Alle	Alle
MsgCompList	MQXR_INIT	Alle	Alle
ChannelMonitoring	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Alle	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Alle	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Alle	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP - Kanalexitparameter

Die MQCXP-Struktur wird an jeden Exittyp übergeben, der von einem Nachrichtenkanalagenten (MCA), einem Clientverbindungskanal oder einem Serververbindungskanal aufgerufen wird.

Siehe MQ_CHANNEL_EXIT.

Die in den folgenden Beschreibungen unter 'Exiteingabe' aufgeführten Felder werden vom Kanal ignoriert, sobald der Exit die Kontrolle an den Kanal zurückgibt. Eingabefelder im Parameterblock mit den Kanalexits, die durch den Exit geändert werden, bleiben für seinen nächsten Aufruf nicht erhalten. Änderungen an Eingabe- oder Ausgabefeldern (z. B. am Feld *ExitUserArea*) bleiben nur für Aufrufe der gleichen Exitinstanz erhalten. Diese Änderungen können nicht zur Übergabe von Daten zwischen verschiedenen Exits des gleichen Kanals oder zwischen dem gleichen Exit verschiedener Kanäle verwendet werden.

Zugehörige Verweise

„Felder” auf Seite 1100
In diesem Kapitel werden alle Felder der MQCXP-Struktur aufgeführt und beschrieben.

„Deklaration in Programmiersprache C” auf Seite 1111
Deklaration der MQCXP-Struktur in C

„COBOL-Delaration” auf Seite 1112
Deklaration der MQCXP-Struktur in COCOL

„Deklaration in RPG (ILE)” auf Seite 1113
Deklaration der MQCXP-Struktur in RPG

„Deklaration in System/390 Assembler” auf Seite 1114
Deklaration der MQCXP-Struktur in System/390-Assembler

Felder

In diesem Kapitel werden alle Felder der MQCXP-Struktur aufgeführt und beschrieben.

StrucId (MQCHAR4)
Dieses Feld gibt die Struktur-ID an.

Folgende Werte sind möglich:

MQCXP_STRUC_ID
ID der Kanalexitparameterstruktur.

Für die Programmiersprache C ist auch die Konstante MQCXP_STRUC_ID_ARRAY definiert. Diese Konstante hat den gleichen Wert wie die Konstante MQCXP_STRUC_ID, es handelt sich dabei jedoch um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Dies ist ein Eingabefeld für den Exit.

Version (MQLONG)
Dieses Feld gibt die Strukturversionsnummer an.

Der Wert hängt von der Umgebung ab:

MQCXP_VERSION_1
Parameterstruktur des Kanalexits für Version-1.

MQCXP_VERSION_2
Parameterstruktur des Kanalexits für Version-2.

Das Feld hat diesen Wert in den folgenden Umgebungen: HP Integrity NonStop Server.

MQCXP_VERSION_3
Parameterstruktur des Kanalexits für Version-3.

Dieses Feld hat diesen Wert in den folgenden Umgebungen: UNIX-Systeme, die sonst nirgendwo aufgeführt sind.

MQCXP_VERSION_4
Parameterstruktur des Kanalexits für Version-4.

MQCXP_VERSION_5
Parameterstruktur des Kanalexits für Version-5.

MQCXP_VERSION_6
Parameterstruktur des Kanalexits für Version-6.

MQCXP_VERSION_8
Parameterstruktur des Kanalexits für Version-8.

Das Feld hat diesen Wert in den folgenden Umgebungen: z/OS, AIX, HP-UX, Linux, IBM i, Solaris, Windows.

Felder, die nur in den aktuelleren Versionen der Struktur vorliegen, sind in den Beschreibungen der Felder als solche angegeben. Die folgende Konstante definiert die Nummer der aktuellen Version:

MQCXP_CURRENT_VERSION
Aktuelle Version der Parameterstruktur des Kanalexits.

Der Wert hängt von der Umgebung ab.

Anmerkung: Wenn eine neue Version der MQCXP-Struktur eingeführt wird, wird der Aufbau der bestehenden Komponente nicht geändert. Der Exit muss daher sicherstellen, dass die Versionsnummer größergleich der niedrigsten Version ist, die die Felder enthält, welche der Exit verwenden muss.

Dies ist ein Eingabefeld für den Exit.

ExitId (MQLONG)
Dieses Feld gibt an, welcher Exittyp aufgerufen wird, und wird beim Zugriff auf die Exitroutine festgelegt.

Folgende Werte sind möglich:

MQXT_CHANNEL_SEC_EXIT
Kanalsicherheitsexit.

MQXT_CHANNEL_MSG_EXIT
Kanalnachrichtenexit.

MQXT_CHANNEL_SEND_EXIT
Kanalsendeexit.

MQXT_CHANNEL_RCV_EXIT
Kanalempfangsexit.

MQXT_CHANNEL_MSG_RETRY_EXIT
Exit für Nachrichtenwiederholungen.

MQXT_CHANNEL_AUTO_DEF_EXIT
Exit für die automatische Kanaldefinition.

Unter z/OS wird dieser Exittyp nur für Kanäle vom Typ MQCHT_CLUSSDR und MQCHT_CLUSRCVR unterstützt.

Dies ist ein Eingabefeld für den Exit.

ExitReason (MQLONG)
Dieses Feld gibt an, weshalb der Exit aufgerufen wird, und wird beim Zugriff auf die Exitroutine festgelegt.

Es wird nicht vom Exit für die automatische Definition verwendet. Folgende Werte sind möglich:

MQXR_INIT
Exitinitialisierung.

Dieser Wert gibt an, dass der Exit zum ersten Mal aufgerufen wird. Er ermöglicht dem Exit, alle Ressourcen, die er benötigt, anzufordern und zu initialisieren (z. B. Speicher).

MQXR_TERM
Exitbeendigung.

Dieser Wert gibt an, dass der Exit in Kürze beendet wird. Der Exit sollte alle Ressourcen freigeben, die er seit seiner Initialisierung angefordert hat (z. B. Speicher).

MQXR_MSG
Verarbeiten einer Nachricht.

Dieser Wert gibt an, dass der Exit zum Verarbeiten einer Nachricht aufgerufen wird. Dieser Wert tritt nur bei Kanalnachrichtenexits auf.

MQXR_XMIT
Verarbeiten einer Übertragung.

Dieser Wert tritt nur bei Kanalsende- und Kanalempfangsexits auf.

MQXR_SEC_MSG
Sicherheitsnachricht erhalten.

Dieser Wert tritt nur bei Kanalsicherheitsexits auf.

MQXR_INIT_SEC
Sicherheitsaustausch initialisieren.

Dieser Wert tritt nur bei Kanalsicherheitsexits auf.

Der Sicherheitsexit des Empfängers wird immer mit dieser Ursache aufgerufen, und zwar unmittelbar, nachdem er mit MQXR_INIT aufgerufen wurde, damit er einen Sicherheitsaustausch initialisieren kann. Wenn er die Gelegenheit ablehnt (durch Rückgabe von MQXCC_OK an Stelle von MQXCC_SEND_SEC_MSG oder MQXCC_SEND_AND_REQUEST_SEC_MSG), wird der Sicherheitsexit des Absenders mit MQXR_INIT_SEC aufgerufen.

Wenn der Sicherheitsexit des Empfängers einen Sicherheitsaustausch initialisiert (durch Rückgabe von MQXCC_SEND_SEC_MSG oder MQXCC_SEND_AND_REQUEST_SEC_MSG), wird der Sicherheitsexit des Absenders nie mit MQXR_INIT_SEC, sondern mit MQXR_SEC_MSG aufgerufen, um die Nachricht des Empfängers zu verarbeiten. (In beiden Fällen wird er zuerst mit MQXR_INIT aufgerufen).

Wenn keiner der Sicherheitsexits die Beendigung des Kanals anfordert (durch Setzen von *ExitResponse* auf MQXCC_SUPPRESS_FUNCTION oder MQXCC_CLOSE_CHANNEL), muss der Sicherheitsaustausch auf der Seite abgeschlossen werden, auf der der Austausch initialisiert wurde. Wenn also ein Sicherheitsexit mit MQXR_INIT_SEC aufgerufen wird und einen Austausch initialisiert, wird der Exit beim nächsten Mal mit MQXR_SEC_MSG aufgerufen. Dies geschieht unabhängig davon, ob eine Sicherheitsnachricht für den Exit verarbeitet werden muss oder nicht. Eine Sicherheitsnachricht wird nur ausgegeben, wenn der Partner MQXCC_SEND_SEC_MSG oder MQXCC_SEND_AND_REQUEST_SEC_MSG zurückgibt, nicht jedoch, wenn der Partner MQXCC_OK zurückgibt oder beim Partner kein Sicherheitsexit vorhanden ist. Liegt keine Sicherheitsnachricht zum Verarbeiten vor, wird der Sicherheitsexit auf der initialisierenden Seite erneut mit einer *DataLength* von 0 aufgerufen.

MQXR_RETRY
Eine Nachricht wiederholen.

Dieser Wert tritt nur bei Exits für Nachrichtenwiederholungen auf.

MQXR_AUTO_CLUSSDR
Automatische Definition eines Clustersenderkanals.

Dieser Wert tritt nur bei Exits für die automatische Kanaldefinition auf.

MQXR_AUTO_RECEIVER
Automatische Definition eines Empfängerkanals.

Dieser Wert tritt nur bei Exits für die automatische Kanaldefinition auf.

MQXR_AUTO_SVRCONN
Automatische Definition eines Serververbindungskanals.

Dieser Wert tritt nur bei Exits für die automatische Kanaldefinition auf.

MQXR_AUTO_CLUSRCVR
Automatische Definition eines Clusterempfängerkanals.

Dieser Wert tritt nur bei Exits für die automatische Kanaldefinition auf.

MQXR_SEC_PARMS
Sicherheitsparameter

Dieser Wert gilt nur für Sicherheitsexits und zeigt an, dass eine MQCSP-Struktur an den Exit übergeben wird. Weitere Informationen finden Sie im Abschnitt „MQCSP - Sicherheitsparameter" auf Seite 315.

Anmerkung:

1. Wenn Sie mehrere Exits für einen Kanal definiert haben, werden sie beim Initialisieren des Nachrichtenkanalagenten einzeln mit MQXR_INIT aufgerufen. Beim Beenden des Nachrichtenkanalagenten werden sie jeweils mit MQXR_TERM aufgerufen.
2. Beim Exit für die automatische Kanaldefinition wird kein *ExitReason* gesetzt, wenn die *Version* niedriger ist als MQCXP_VERSION_4. Stattdessen wird der Wert MQXR_AUTO_SVRCONN impliziert.

Dies ist ein Eingabefeld für den Exit.

ExitResponse (MQLONG)
Dieses Feld gibt die Antwort des Exits an.

Dieses Feld wird vom Exit zur Kommunikation mit dem Nachrichtenkanalagent eingerichtet.Folgende Werte sind zulässig:

MQXCC_OK
Exit erfolgreich ausgeführt.

- Beim Kanalsicherheitsexit zeigt dieser Wert an, dass die Nachrichtenübertragung nun normal fortgesetzt werden kann.
- Beim Exit für Kanalnachrichtenwiederholungen zeigt dieser Wert an, dass der Nachrichtenkanalagent für die Dauer des vom Exit im Feld *MsgRetryInterval* in MQCXP zurückgegebenen Zeitintervalls warten muss und anschließend die Nachricht erneut wiederholt.

Das Feld *ExitResponse2* enthält unter Umständen weitere Informationen.

MQXCC_SUPPRESS_FUNCTION
Funktion unterdrücken.

- Beim Kanalsicherheitsexit zeigt dieser Wert an, dass der Kanal beendet werden muss.
- Beim Kanalnachrichtenexit zeigt dieser Wert an, dass die Nachricht nicht weiter an ihr Ziel geleitet wird. Stattdessen generiert der Nachrichtenkanalagent eine Abweichungsberichtsnachricht (falls vom Absender der Originalnachricht angefordert) und stellt die im Originalpuffer enthaltene Nachricht in die Warteschlange für nicht zustellbare Nachrichten (wenn der Absender MQRO_DEAD_LETTER_Q angegeben hat) bzw. verwirft sie (wenn der Absender MQRO_DISCARD_MSG angegeben hat).

  Wenn der Absender bei persistenten Nachrichten MQRO_DEAD_LETTER_Q angegeben hat, die Nachricht jedoch nicht in die Warteschlange für nicht zustellbare Nachrichten eingereiht werden kann bzw. keine Warteschlange für nicht zustellbare Nachrichten vorhanden ist, bleibt die Originalnachricht in der Übertragungswarteschlange und es wird keine Berichtsnachricht generiert. Die Originalnachricht bleibt zudem auch in der Übertragungswarteschlange, wenn die Berichtsnachricht nicht erfolgreich generiert werden kann.

  Das Feld *Feedback* in der MQDLH-Struktur am Anfang der Nachricht in der Warteschlange für nicht zustellbare Nachrichten zeigt an, weshalb die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Dieser Rückkopplungscode wird auch im Nachrichtendeskriptor der Abweichungsberichtsnachricht verwendet (falls vom Absender angefordert).
- Beim Exit für Kanalnachrichtenwiederholungen zeigt dieser Wert an, dass der Nachrichtenkanalagent nicht wartet und die Nachricht nicht wiederholt. Stattdessen setzt er seine normale Fehlerverarbeitung unverzüglich fort (je nach Angabe des Absenders der Nachricht wird die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht oder verworfen).
- Beim Exit für die automatische Kanaldefinition muss entweder MQXCC_OK oder MQXCC_SUPPRESS_FUNCTION angegeben werden. Ist keiner dieser Werte angegeben, wird standardmäßig der

Wert MQXCC_SUPPRESS_FUNCTION vorausgesetzt und die automatische Definition wird abgebrochen.

Diese Antwort wird bei Kanalsende- und Kanalempfangsexits nicht unterstützt.

MQXCC_SEND_SEC_MSG
Sicherheitsnachricht senden.

Dieser Wert kann nur von einem Kanalsicherheitsexit gesetzt werden. Er zeigt an, dass der Exit eine Sicherheitsnachricht bereitgestellt hat, die an den Partner übertragen werden muss.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Sicherheitsnachricht senden, die eine Antwort erfordert.

Dieser Wert kann nur von einem Kanalsicherheitsexit gesetzt werden. Er zeigt an,

- dass der Exit eine Sicherheitsnachricht bereitgestellt hat, die an den Partner übertragen werden muss, und
- dass der Exit eine Antwort des Partners erfordert. Wird keine Antwort empfangen, muss der Kanal beendet werden, da der Exit noch nicht entschieden hat, ob die Kommunikation fortgesetzt werden kann.

MQXCC_SUPPRESS_EXIT
Exit unterdrücken.

- Dieser Wert kann von allen Kanalexittypen mit Ausnahme des Sicherheitsexits oder Exits zur automatischen Definition gesetzt werden. Er unterdrückt das weitere Aufrufen dieses Exits (gleichbedeutend mit einem leeren Namen in der Kanaldefinition) bis zur Beendigung des Kanals, wenn der Exit wieder mit einem *ExitReason* MQXR_TERM aufgerufen wird.
- Wenn ein Exit für Nachrichtenwiederholungen diesen Wert zurückgibt, werden die Nachrichtenwiederholungen für nachfolgende Nachrichten wie gehabt durch die Kanalattribute *MsgRetryCount* und *MsgRetryInterval* gesteuert. Bei der aktuellen Nachricht führt der Nachrichtenkanalagent die Anzahl der ausstehenden Wiederholungen in den durch das Kanalattribut *MsgRetryInterval* vorgegebenen Intervallen aus. Dies geschieht jedoch nur, wenn ein Ursachencode vorliegt, den der Nachrichtenkanalagent normalerweise wiederholen würde (siehe Beschreibung des Feldes *MsgRetryCount* unter „MQCD - Kanaldefinition” auf Seite 1058). Die Anzahl der ausstehenden Wiederholungen entspricht dem Wert des Attributs *MsgRetryCount* minus der Anzahl der Rückgaben des Werts MQXCC_OK durch den Exit für die aktuelle Nachricht. Ist diese Zahl negativ, führt der Nachrichtenkanalagent keine weiteren Wiederholungen der aktuellen Nachricht durch.

MQXCC_CLOSE_CHANNEL
Kanal schließen.

Dieser Wert kann von allen Kanalexittypen mit Ausnahme des Exits zur automatischen Definition gesetzt werden.

Wenn die gemeinsame Nutzung von Datenaustausch nicht aktiviert ist, schließt dieser Wert den Kanal.

Ist die gemeinsame Nutzung von Datenaustausch aktiviert, beendet dieser Wert den Datenaustausch. Wenn es sich bei diesem Datenaustausch um den einzigen Datenaustausch auf dem Kanal handelt, wird der Kanal ebenfalls geschlossen.

Dieses Feld ist ein Ein-/Ausgabefeld im Exit.

ExitResponse2 (MQLONG)
Dieses Feld gibt die sekundäre Antwort des Exits an.

Dieses Feld wird beim Zugriff auf die Exitroutine auf Null gesetzt.Es kann vom Exit festgelegt werden, um den WebSphere MQ-Kanalfunktionen weitere Informationen zu liefern. Es wird nicht vom Exit für die automatische Definition verwendet.

Der Exit kann mindestens einen der folgenden Werte festlegen. Wenn mehrere Werte erforderlich sind, werden sie hinzugefügt. Auf ungültige Kombinationen wird hingewiesen, andere Kombinationen sind zulässig.

MQXR2_PUT_WITH_DEF_ACTION
Einreihen mit Standardaktion.

Dieser Wert wird vom Kanalnachrichtenexit des Empfängers festgelegt. Er gibt an, dass die Nachricht mit der Standardaktion des Nachrichtenkanalagenten einzustellen ist, bei der es sich entweder um die Standard-Benutzer-ID des Nachrichtenkanalagenten oder um die *UserIdentifier* (Benutzer-ID) des Kontextes im MQMD (Nachrichtendeskriptor) der Nachricht handelt.

Der Wert ist 0, was dem Anfangswert entspricht, der beim Aufrufen des Exits gesetzt wird. Die Konstante wird zu Dokumentationszwecken angegeben.

MQXR2_PUT_WITH_DEF_USERID
Einreihen mit Standard-Benutzer-ID.

Dieser Wert kann nur vom Kanalnachrichtenexit des Empfängers festgelegt werden. Er gibt an, dass die Nachricht mit der Standard-Benutzer-ID des Nachrichtenkanalagenten einzustellen ist.

MQXR2_PUT_WITH_MSG_USERID
Einreihen mit Benutzer-ID der Nachricht.

Dieser Wert kann nur vom Kanalnachrichtenexit des Empfängers festgelegt werden. Er gibt an, dass die Nachricht mit dem/der *UserIdentifier* (Benutzer-ID) des Kontextes im MQMD (Nachrichtendeskriptor) der Nachricht einzustellen ist (wurde möglicherweise vom Exit verändert).

Es darf nur einer der Werte MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERID und MQXR2_PUT_WITH_MSG_USERID gesetzt werden.

MQXR2_USE_AGENT_BUFFER
Agentenpuffer verwenden.

Dieser Wert gibt an, dass sich alle weiterzugebenden Daten im Agentenpuffer (*AgentBuffer*) befinden, nicht in der Exitpufferadresse (*ExitBufferAddr*).

Der Wert ist 0, was dem Anfangswert entspricht, der beim Aufrufen des Exits gesetzt wird. Die Konstante wird zu Dokumentationszwecken angegeben.

MQXR2_USE_EXIT_BUFFER
Exitpuffer verwenden.

Dieser Wert gibt an, dass sich alle weiterzugebenden Daten im Agentenpuffer (*ExitBufferAddr*) befinden, nicht in der Exitpufferadresse (*AgentBuffer*).

Es darf nur einer der Werte MQXR2_USE_AGENT_BUFFER und MQXR2_USE_EXIT_BUFFER gesetzt werden.

MQXR2_DEFAULT_CONTINUATION
Standardfortsetzung.

Die Fortsetzung mit dem nächsten Exit in der Kette hängt von der Antwort des zuletzt aufgerufenen Exits ab:

- Wenn MQXCC_SUPPRESS_FUNCTION oder MQXCC_CLOSE_CHANNEL zurückgegeben wird, werden keine weiteren Exits in der Kette aufgerufen.
- Wenn nicht, wird der nächste Exit in der Kette aufgerufen.

MQXR2_CONTINUE_CHAIN
Fortsetzung mit dem nächsten Exit.

MQXR2_SUPPRESS_CHAIN
Die übrigen Exits in der Kette werden übersprungen.

Dies ist ein Ein-/Ausgabefeld für den Exit.

Feedback (MQLONG)
Dieses Feld gibt den Rückkopplungscode an.

Dieses Feld wird beim Zugriff auf die Exitroutine auf MQFB_NONE gesetzt.

Wenn ein Kanalnachrichtenexit das Feld *ExitResponse* auf den Wert MQXCC_SUPPRESS_FUNCTION setzt, gibt das Feld *Feedback* den Rückkopplungscode an, der anzeigt, weshalb die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Er wird zudem zum Senden eines Abweichungsberichts verwendet, falls dieser angefordert wurde. In diesem Fall wird folgender Rückkopplungscode verwendet, wenn das Feld *Feedback* den Wert MQFB_NONE hat:

MQFB_STOPPED_BY_MSG_EXIT
Nachricht vom Kanalnachrichtenexit gestoppt.

Der in diesem Feld von den Kanalsicherheits-, Sende- und Empfangsexits sowie von den Exits für Nachrichtenwiederholung zurückgegebene Wert wird nicht vom Nachrichtenkanalagenten verwendet.

Der in diesem Feld von den Exits für die automatische Definition zurückgegebene Wert wird nicht verwendet, wenn *ExitResponse* den Wert MQXCC_OK hat. Andernfalls wird er für den Parameter *AuxErrorDataInt1* in der Ereignisnachricht verwendet.

Dieses Feld ist ein Ein-/Ausgabefeld im Exit.

MaxSegmentLength (MQLONG)
Dieses Feld gibt die maximale Anzahl in Byte an, die in einer einzelnen Übertragung gesendet werden kann.

Es wird nicht vom Exit für die automatische Definition verwendet. Für einen Kanalsendeexit ist es von Bedeutung, da dieser Exit sicherstellen muss, dass die Größe eines Übertragungssegments nicht den Wert von *MaxSegmentLength* überschreitet. Die Länge beinhaltet auch die ersten 8 Byte, die der Exit nicht ändern darf. Der Wert wird bei Initialisierung des Kanals zwischen den WebSphere MQ-Kanalfunktionen verhandelt. Weitere Informationen zur Länge von Segmenten finden Sie unter Kanalexitprogramme schreiben.

Der Wert in diesem Feld ist nicht aussagefähig, wenn *ExitReason* den Wert MQXR_INIT hat.

Dies ist ein Eingabefeld für den Exit.

ExitUserArea (MQBYTE16)
Dieses Feld gibt den Exitbenutzerbereich an, ein Feld, das zur Verwendung durch den Exit zur Verfügung steht.

Es wird vor dem ersten Aufruf des Exits (dessen *ExitReason* den Wert MQXR_INIT hat) mit einer binären Null initialisiert. Alle Änderungen, die der Exit danach am Feld vornimmt, werden bei den Exitaufrufen beibehalten.

Der folgende Wert ist definiert:

MQXUA_NONE
Keine Benutzerinformationen.

Der Wert ist eine binäre Null für die Feldlänge.

Für die Programmiersprache C ist auch die Konstante MQXUA_NONE_ARRAY definiert. Diese Konstante hat den gleichen Wert wie die Konstante MQXUA_NONE, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Die Länge dieses Felds wird durch MQ_EXIT_USER_AREA_LENGTH angegeben. Dies ist ein Ein-/Ausgabefeld für den Exit.

ExitData (MQCHAR32)
Dieses Feld gibt die Exitdaten an.

Dieses Feld wird beim Zugriff auf die Exitroutine auf Informationen gesetzt, welche die WebSphere MQ-Kanalfunktionen aus der Kanaldefinition abgerufen haben.Stehen solche Informationen nicht zur Verfügung, enthält dieses Feld nur Leerzeichen.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

Dies ist ein Eingabefeld für den Exit.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_2.

MsgRetryCount (MQLONG)
Dieses Feld gibt an, wie oft die Nachricht wiederholt wurde.

Beim ersten Aufruf des Exits für eine bestimmte Nachricht hat dieses Feld den Wert 0 (es wurden noch keine Wiederholungen versucht).Bei jedem nachfolgenden Aufruf des Exits für diese Nachricht wird der Wert durch den Nachrichtenkanalagenten um 1 erhöht.

Dies ist ein Eingabefeld für den Exit. Der Wert in diesem Feld ist nicht aussagefähig, wenn *ExitReason* den Wert MQXR_INIT hat. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_2.

MsgRetryInterval (MQLONG)
Dieses Feld gibt das Mindestintervall in Millisekunden an, nach dem eine PUT-Operation wiederholt wird.

Beim ersten Aufruf des Exits für eine bestimmte Nachricht enthält dieses Feld den Wert des Kanalattributs *MsgRetryInterval*.Der Exit kann den Wert unverändert lassen oder ihn so ändern, dass ein anderes Zeitintervall in Millisekunden angegeben wird. Wenn der Exit den Wert MQXCC_OK in der *ExitResponse* zurückgibt, wartet der Nachrichtenkanalagent mindestens für die Dauer dieses Zeitintervalls, bevor er die MQOPEN- bzw. MQPUT-Operation wiederholt. Das angegebene Zeitintervall muss Null oder größer sein.

Beim zweiten Aufruf sowie allen darauffolgenden Aufrufen des Exits für die Nachricht enthält dieses Feld den Wert, der beim vorherigen Aufruf des Exits zurückgegeben wurde.

Wenn der im Feld *MsgRetryInterval* zurückgegebene Wert kleiner als Null oder größer als 999999999 ist und *ExitResponse* den Wert MQXCC_OK hat, ignoriert der Nachrichtenkanalagent das Feld *MsgRetryInterval* in MQCXP und wartet stattdessen für die Dauer des vom Kanalattribut *MsgRetryInterval* festgelegten Intervalls.

Dies ist ein Ein-/Ausgabefeld für den Exit. Der Wert in diesem Feld ist nicht aussagefähig, wenn *ExitReason* den Wert MQXR_INIT hat. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_2.

MsgRetryReason (MQLONG)
Dieses Feld gibt den Ursachencode des vorherigen Versuchs, die Nachricht einzureihen, an.

Dieses Feld ist der Ursachencode des vorherigen Versuchs, die Nachricht einzureihen. Es enthält einen der MQRC_*-Werte.

Dies ist ein Eingabefeld für den Exit. Der Wert in diesem Feld ist nicht aussagefähig, wenn *ExitReason* den Wert MQXR_INIT hat. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_2.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

HeaderLength (MQLONG)
Dieses Feld gibt die Länge der Kopfzeileninformationen an.

Dieses Feld ist nur für einen Nachrichtenexit und einen Exit für Nachrichtenwiederholung von Bedeutung.Der Wert ist die Länge der Routing-Header-Strukturen am Anfang der Nachrichtendaten. Dazu gehören die MQXQH-Struktur, der MQMDE (Header der Nachrichtenbeschreibungserweiterung) und (bei Verteilerlistennachrichten) die MQDH-Struktur sowie Gruppen von MQOR- und MQPMR-Datensätzen, die der MQXQH-Struktur folgen.

Der Nachrichtenexit kann diese Kopfzeileninformationen prüfen und bei Bedarf ändern, doch die Daten, die der Exit zurückgibt, müssen weiterhin das richtige Format haben. Der Exit darf die Headerdaten beispielsweise auf der Sendeseite nicht verschlüsseln oder komprimieren, selbst wenn der Nachrichtenexit auf der Empfangsseite ausgleichende Änderungen vornimmt.

Wenn der Nachrichtenexit zum Beispiel die Länge der Kopfzeileninformationen ändert (beispielsweise durch Hinzufügen einer weiteren Zieladresse zu einer Verteilerlistennachricht), muss er den Wert der Headerlänge (*HeaderLength*) entsprechend vor der Rückgabe ändern.

Dies ist ein Ein-/Ausgabefeld für den Exit. Der Wert in diesem Feld ist nicht aussagefähig, wenn *ExitReason* den Wert MQXR_INIT hat. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

PartnerName (MQCHAR48)
Dieses Feld gibt den Namen des Partners an.

Der Name des Partners lautet wie folgt:

- Bei SVRCONN-Kanälen ist es die ID des am Client angemeldeten Benutzers.
- Bei allen anderen Kanaltypen ist es der Warteschlangenmanagername des Partners.

Bei Initialisierung des Exits ist dieses Feld leer, da der Warteschlangenmanager den Namen des Partners erst nach Abschluss der ersten Initialisierung kennt.

Dies ist ein Eingabefeld für den Exit. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

FAPLevel (MQLONG)
Verhandelte Format- und Protokollebenen.

Dies ist ein Eingabefeld für den Exit. Dieses Feld darf nur nach Anweisung des IBM Service geändert werden.Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

CapabilityFlags (MQLONG)
Dieses Feld gibt die Funktionsflags an.

Folgendes ist definiert:

MQCF_NONE
Keine Flags.

MQCF_DIST_LISTS
Unterstützte Verteilerlisten.

Dies ist ein Eingabefeld für den Exit. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

ExitNumber (MQLONG)
Dieses Feld gibt die Ordinalzahl des Exits an.

Die Ordinalzahl des Exits innerhalb des in *ExitId* definierten Typs.Beispiel: Wenn der Exit, der aufgerufen wird, der dritte definierte Nachrichtenexit ist, enthält dieses Feld den Wert 3. Wenn es sich bei dem Exittyp um einen Typ handelt, für den keine Liste der Exits definiert werden kann (z. B. ein Sicherheitsexit), hat dieses Feld den Wert 1.

Dies ist ein Eingabefeld für den Exit. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_3.

Die folgenden Felder in dieser Struktur sind nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_5.

ExitSpace (MQLONG)
Dieses Feld gibt die Anzahl der Byte im Übertragungspuffer an, der zur Verwendung durch den Exit reserviert ist.

Dieses Feld ist nur für Sendeexits von Bedeutung.Es gibt die Speichermenge in Byte an, die die WebSphere MQ-Kanalfunktionen im Übertragungspuffer reservieren, den der Exit verwendet. Dieses Feld ermöglicht dem Exit, dem Übertragungspuffer eine kleine Datenmenge hinzuzufügen (i.d.R. nicht mehr als einige Hundert Byte), die ein komplementärer Empfangsexit auf der anderen Seite verwendet. Die vom Sendeexit hinzugefügten Daten müssen vom Empfangsexit entfernt werden.

Der Wert ist unter z/OSimmer null.

Anmerkung: Diese Funktion darf nicht zum Senden großer Datenmengen verwendet werden, da dies die Leistung beeinträchtigen oder sogar den Betrieb des Kanals blockieren kann.

Durch Festlegen von *ExitSpace* wird sichergestellt, dass dem Exit immer mindestens diese Anzahl von Byte im Übertragungspuffer zur Verwendung zur Verfügung steht. Der Exit kann jedoch weniger bzw. mehr als die reservierte Anzahl verwenden, wenn im Übertragungspuffer Speicherplatz zur Verfügung steht. Der Exitspeicher im Puffer wird nach den bestehenden Daten bereitgestellt.

ExitSpace kann nur vom Exit festgelegt werden, wenn *ExitReason* den Wert MQXR_INIT hat. In allen anderen Fällen wird der vom Exit zurückgegebene Wert ignoriert. Bei der Eingabe für den Exit hat *ExitSpace* den Wert 0 für den MQXR_INIT-Aufruf. In allen anderen Fällen hat er den Wert, der vom MQXR_INIT-Aufruf zurückgegeben wurde.

Wenn der vom MQXR_INIT-Aufruf zurückgegebene Wert negativ ist oder nach Reservierung des angeforderten Exitspeicherplatzes im Übertragungspuffer weniger als 1024 Byte für Nachrichtendaten für alle Sendeexits in der Kette zur Verfügung stehen, gibt der Nachrichtenkanalagent eine Fehlermeldung aus und schließt den Kanal. Der Nachrichtenkanalagent gibt auch dann eine Fehlermeldung aus und schließt den Kanal, wenn die Exits in der Sendeexitkette während der Datenübertragung mehr Benutzeradressbereich zuweisen, als sie reserviert haben, sodass weniger als 1024 Byte für Nachrichtendaten im Übertragungspuffer für Nachrichtendaten zur Verfügung stehen. Der Grenzwert von 1024 ermöglicht die Verarbeitung der Steuerungs- und Verwaltungsabläufe des Kanals durch die Kette der Sendeexits, ohne dass Abläufe segmentiert werden müssen.

Dies ist ein Ein-/Ausgabefeld für den Exit, wenn *ExitReason* den Wert MQXR_INIT hat. In allen anderen Fällen ist dieses Feld ein Eingabefeld. Das Feld ist nicht vorhanden, wenn die *Version* niedriger ist als MQCXP_VERSION_5.

SSLCertUserId (MQCHAR12)
Dieses Feld gibt die dem fernen Zertifikat zugeordnete Benutzer-ID an.

Mit Ausnahme von z/OS ist es auf allen Plattformen leer.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist.

SSLRemCertIssNameLength (MQLONG)
Dieses Feld gibt die Länge in Byte des vollständig definierten Namens des Ausstellers des fernen Zertifikats an, auf das "SSLCertRemoteIssuerNamePtr" verweist.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist. Der Wert ist null, wenn es sich nicht um einen SSL-Kanal handelt.

SSLRemCertIssNamePtr (PMQVOID)
Dieses Feld gibt die Adresse des vollständig definierten Namens des Ausstellers des fernen Zertifikats an.

Sein Wert ist ein Nullzeiger, sofern es sich nicht um einen SSL-Kanal handelt.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist.

Anmerkung: Das Verhalten von Kanalsicherheitsexits bei der Ermittlung des definierten Namens des Zertifikatinhabers und des Zertifikatausstellers hat sich ab Version WebSphere MQ 7.1 geändert. Weitere Informationen finden Sie im Abschnitt Kanalsicherheits-Exitprogramme.

SecurityParms (PMQCSP)
Dieses Feld gibt die Adresse der MQSCP-Struktur an, mit der eine Benutzer-ID und ein Kennwort angegeben werden.

Der Anfangswert dieses Felds ist der Nullzeiger.

Dies ist ein Ein-/Ausgabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist.

CurHdrCompression (MQLONG)
Dieses Feld gibt an, mit welcher Technik die Headerdaten aktuell komprimiert werden.

Es ist auf einen der folgenden Werte gesetzt:

MQCOMPRESS_NONE
Es werden keine Headerdaten komprimiert.

MQCOMPRESS_SYSTEM
Headerdaten werden komprimiert.

Der Wert kann geändert werden, indem Sie den Nachrichtenexit eines Kanals an einen der verhandelten, unterstützten Werte senden, auf den aus dem Feld "HdrCompList" im MQCD zugegriffen wird. Dies ermöglicht die Auswahl des Komprimierungsverfahrens für die Headerdaten jeder Nachricht auf Grundlage des jeweiligen Nachrichteninhalts. Der geänderte Wert wird nur für die aktuelle Nachricht verwendet. Der Kanal wird beendet, wenn das Attribut in einen nicht unterstützten Wert geändert wird. Der Wert wird ignoriert, wenn er außerhalb des Nachrichtenexits eines Kanals geändert wird.

Dies ist ein Ein-/Ausgabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist.

CurMsgCompression (MQLONG)
Dieses Feld gibt an, mit welcher Technik die Nachrichtendaten aktuell komprimiert werden.

Es ist auf einen der folgenden Werte gesetzt:

MQCOMPRESS_NONE
Es werden keine Headerdaten komprimiert.

MQCOMPRESS_RLE
Nachrichtendaten werden mittels Lauflängencodierung komprimiert.

MQCOMPRESS_ZLIBFAST
Die Komprimierung der Nachrichtendaten erfolgt unter Verwendung der ZLIB-Komprimierungstechnik. Dabei wird eine kurze Komprimierungszeit bevorzugt.

MQCOMPRESS_ZLIBHIGH
Die Komprimierung der Nachrichtendaten erfolgt unter Verwendung der ZLIB-Komprimierungstechnik. Dabei wird eine hohe Komprimierungsstufe bevorzugt.

Der Wert kann geändert werden, indem Sie den Nachrichtenexit eines Kanals an einen der verhandelten, unterstützten Werte senden, auf den aus dem Feld "MsgCompList" im MQCD zugegriffen wird. Dies ermöglicht die Auswahl des Komprimierungsverfahrens für die Nachrichtendaten jeder Nachricht auf Grundlage des jeweiligen Nachrichteninhalts. Der geänderte Wert wird nur für die aktuelle Nachricht verwendet. Der Kanal wird beendet, wenn das Attribut in einen nicht unterstützten Wert geändert wird. Der Wert wird ignoriert, wenn er außerhalb des Nachrichtenexits eines Kanals geändert wird.

Dies ist ein Ein-/Ausgabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_6 ist.

Hconn (MQHCONN)
Dieses Feld gibt die Verbindungskennung an, die der Exit verwendet, wenn er MQI innerhalb des Exits aufrufen muss.

Dieses Feld ist nicht relevant für Exits, die in Clientverbindungskanälen aktiv sind, in denen es den Wert MQHC_UNUSABLE_HCONN (-1) enthält.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_7 ist.

SharingConversations (MQBOOL)
Dieses Feld gibt an, ob nur dieser Dialog derzeit auf dieser Kanalinstanz aktiv sein kann oder ob mehrere Dialoge auf der Kanalinstanz aktiv sein können.

Es gibt ebenfalls an, ob das Exitprogramm dem Risiko unterliegt, dass der MQCD durch ein anderes, gleichzeitig ausgeführtes Exitprogramm geändert wird.

Dieses Feld ist nur für Exitprogramme relevant, die auf Client- oder Serververbindungskanälen aktiv sind.

Es ist auf einen der folgenden Werte gesetzt:

FALSE
Dies ist die einzige Instanz, die derzeit auf dieser Kanalinstanz aktiv sein kann. Dadurch kann der Exit die MQCD-Felder sicher aktualisieren, ohne dass es zu einem Konflikt mit anderen Exits kommt, die auf anderen Kanalinstanzen aktiv sind. Ob Änderungen der MQCD-Felder vom Kanal verarbeitet werden, ist in der Tabelle der MQCD-Felder unter „Ändern von MQCD-Feldern in einem Kanalexit" auf Seite 1097 definiert.

TRUE
Dies ist nicht die einzige Instanz, die derzeit in dieser Kanalinstanz aktiv sein kann. Keine Änderung im MQCD wird vom Kanal verarbeitet mit Ausnahme der Änderungen, die in der Tabelle der MQCD-Felder unter „Ändern von MQCD-Feldern in einem Kanalexit" auf Seite 1097 für andere Exit-Ursachen als MQXR_INIT aufgeführt sind. Wenn dieser Exit die MQCD-Felder aktualisiert, muss sichergestellt werden, dass es zu keinem Konflikt mit anderen Exits kommt, die gleichzeitig in anderen Dialogen aktiv sind, indem die auf dieser Kanalinstanz aktiven Exits seriell verarbeitet werden.

Dies ist ein Eingabefeld für den Exit. Dieses Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_7 ist.

MCAUserSource (MQLONG)
Dieses Feld gibt die Quelle für die bereitgestellte MCA-Benutzer-ID an.

Es kann einen der folgenden Werte enthalten:

MQUSRC_MAP
Die Benutzer-ID, die im Attribut MCAUSER angegeben ist.

MQUSRC_CHANNEL
Die Benutzer-ID wird im Umlaufverfahren vom ankommenden Partner bereitgestellt oder im Feld MCAUSER angegeben, das im Kanalobjekt definiert ist.

Dies ist ein Eingabefeld für den Exit. Das Feld ist nicht vorhanden, wenn 'Version' kleiner als MQCXP_VERSION_8 ist.

pEntryPoints (PMQIEP)
Dieses Feld gibt die Adresse des Schnittstelleneingangspunkts für den MQI- oder DCI-Aufruf an.

Das Feld ist nicht vorhanden, wenn *Version* kleiner als MQCXP_VERSION_8 ist.

Deklaration in Programmiersprache C

Deklaration der MQCXP-Struktur in C

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Type of exit */
  MQLONG    ExitReason;        /* Reason for invoking exit */
  MQLONG    ExitResponse;      /* Response from exit */
  MQLONG    ExitResponse2;     /* Secondary response from exit */
  MQLONG    Feedback;          /* Feedback code */
  MQLONG    MaxSegmentLength;  /* Maximum segment length */
  MQBYTE16  ExitUserArea;      /* Exit user area */
  MQCHAR32  ExitData;          /* Exit data */
  MQLONG    MsgRetryCount;     /* Number of times the message has been
                                  retried */
  MQLONG    MsgRetryInterval;  /* Minimum interval in milliseconds after
                                  which the put operation should be
                                  retried */
  MQLONG    MsgRetryReason;    /* Reason code from previous attempt to
                                  put the message */
  MQLONG    HeaderLength;      /* Length of header information */
  MQCHAR48  PartnerName;       /* Partner Name */
  MQLONG    FAPLevel;          /* Negotiated Formats and Protocols
                                  level */
  MQLONG    CapabilityFlags;   /* Capability flags */
```

```
  MQLONG    ExitNumber;        /* Exit number */
  /* Ver:3 */
  /* Ver:4 */
  MQLONG    ExitSpace;         /* Number of bytes in transmission buffer
                                  reserved for exit to use */
  /* Ver:5 */
  MQCHAR12  SSLCertUserid;     /* User identifier associated
                                  with remote SSL certificate */
  MQLONG    SSLRemCertIssNameLength;   /* Length of
                                  distinguished name of issuer
                                  of remote SSL certificate */
  MQPTR     SSLRemCertIssNamePtr;      /* Address of
                                  distinguished name of issuer
                                  of remote SSL certificate */
  PMQVOID   SecurityParms;     /* Security parameters */
  MQLONG    CurHdrCompression; /* Header data compression
                                  used for current message */
  MQLONG    CurMsgCompression; /* Message data compression
                                  used for current message */
  /* Ver:6 */
  MQHCONN   Hconn;             /* Connection handle */
  MQBOOL    SharingConversations;      /* Multiple conversations
                                  possible on channel inst?  */
  /* Ver:7 */
  MQLONG    MCAUserSource;     /* Source of the provided MCA user ID */
  PMQIEP    pEntryPoints;     /* Address of the MQIEP structure */
}
  /* Ver:8 */
;
```

COBOL-Delaration

Deklaration der MQCXP-Struktur in COCOL

```
**   MQCXP structure
  10 MQCXP.
**    Structure identifier
   15 MQCXP-STRUCID          PIC X(4).
**    Structure version number
   15 MQCXP-VERSION          PIC S9(9) BINARY.
**    Type of exit
   15 MQCXP-EXITID           PIC S9(9) BINARY.
**    Reason for invoking exit
   15 MQCXP-EXITREASON       PIC S9(9) BINARY.
**    Response from exit
   15 MQCXP-EXITRESPONSE     PIC S9(9) BINARY.
**    Secondary response from exit
   15 MQCXP-EXITRESPONSE2    PIC S9(9) BINARY.
**    Feedback code
   15 MQCXP-FEEDBACK         PIC S9(9) BINARY.
**    Maximum segment length
   15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
**    Exit user area
   15 MQCXP-EXITUSERAREA     PIC X(16).
**    Exit data
   15 MQCXP-EXITDATA         PIC X(32).
**    Number of times the message has been retried
   15 MQCXP-MSGRETRYCOUNT    PIC S9(9) BINARY.
**    Minimum interval in milliseconds after which the put operation
**    should be retried
   15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
**    Reason code from previous attempt to put the message
   15 MQCXP-MSGRETRYREASON   PIC S9(9) BINARY.
**    Length of header information
   15 MQCXP-HEADERLENGTH     PIC S9(9) BINARY.
**    Partner Name
   15 MQCXP-PARTNERNAME      PIC X(48).
**    Negotiated Formats and Protocols level
   15 MQCXP-FAPLEVEL         PIC S9(9) BINARY.
**    Capability flags
   15 MQCXP-CAPABILITYFLAGS  PIC S9(9) BINARY.
**    Exit number
   15 MQCXP-EXITNUMBER       PIC S9(9) BINARY.
**    Number of bytes in transmission buffer reserved for exit to use
   15 MQCXP-EXITSPACE        PIC S9(9) BINARY.
**    User Id associated with remote certificate
   15  MQCXP-SSLCERTUSERID   PIC X(12).
** Length of distinguished name of issuer of remote SSL
** certificate
```

```
   15  MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote SSL
** certificate
   15  MQCXP-SSLREMCERTISSNAMEPTR    POINTER.
** Security parameters
   15  MQCXP-SECURITYPARMS           PIC S9(18) BINARY.
** Header data compression used for current message
   15  MQCXP-CURHDRCOMPRESSION       PIC S9(9) BINARY.
** Message data compression used for current message
   15  MQCXP-CURMSGCOMPRESSION       PIC S9(9) BINARY.
** Connection handle
   15 MQCXP-HCONN                    PIC S9(9) BINARY.
** Multiple conversations possible on channel instance?
   15 MQCXP-SHARINGCONVERSATIONS     PIC S9(9) BINARY.
** Source of the provided MCA user ID
   15 MQCXP-MCAUSERSOURCE            PIC S9(9) BINARY.
```

Deklaration in RPG (ILE)

Deklaration der MQCXP-Struktur in RPG

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D* MQCXP Structure
     D*
     D* Structure identifier
     D  CXSID                  1      4
     D* Structure version number
     D  CXVER                  5      8I 0
     D* Type of exit
     D  CXXID                  9     12I 0
     D* Reason for invoking exit
     D  CXREA                 13     16I 0
     D* Response from exit
     D  CXRES                 17     20I 0
     D* Secondary response from exit
     D  CXRE2                 21     24I 0
     D* Feedback code
     D  CXFB                  25     28I 0
     D* Maximum segment length
     D  CXMSL                 29     32I 0
     D* Exit user area
     D  CXUA                  33     48
     D* Exit data
     D  CXDAT                 49     80
     D* Number of times the message has been retried
     D  CXMRC                 81     84I 0
     D* Minimum interval in milliseconds after which the put operation
     D* should be retried
     D  CXMRI                 85     88I 0
     D* Reason code from previous attempt to put the message
     D  CXMRR                 89     92I 0
     D* Length of header information
     D  CXHDL                 93     96I 0
     D* Partner Name
     D  CXPNM                 97    144
     D* Negotiated Formats and Protocols level
     D  CXFAP                145    148I 0
     D* Capability flags
     D  CXCAP                149    152I 0
     D* Exit number
     D  CXEXN                153    156I 0
     D* Number of bytes in transmission buffer reserved for exit to use
     D  CXHDL                157    160I 0
     D* User identifier associated with remote SSL certificate
     D  CXSSLCU              161    172
     D* Length of distinguished name of issuer of remote SSL certificate
     D  CXSRCINL             173    176I 0
     D* Address of distinguished name of issuer of remote SSL certificate
     D  CXSRCINP             177    192*
     D* Security parameters
     D  CXSECP               193    208*
     D* Header data compression used for current message
     D  CXCHC                209    212I 0
     D* Message data compression used for current message
     D  CXCMC                213    216I 0
     D* Connection handle
     D  CXHCONN               217    220I   0
     D* Multiple conversations possible on channel instance?
```

```
     D  CXSHARECONV           221    224I   0
     D* Source of the provided MCA user ID
     D  MCAUSERSOURCE         225    228I   0
```

Deklaration in System/390 Assembler

Deklaration der MQCXP-Struktur in System/390-Assembler

```
MQCXP                   DSECT
MQCXP_STRUCID           DS   CL4   Structure identifier
MQCXP_VERSION           DS   F     Structure version number
MQCXP_EXITID            DS   F     Type of exit
MQCXP_EXITREASON        DS   F     Reason for invoking exit
MQCXP_EXITRESPONSE      DS   F     Response from exit
MQCXP_EXITRESPONSE2     DS   F     Secondary response from exit
MQCXP_FEEDBACK          DS   F     Feedback code
MQCXP_MAXSEGMENTLENGTH  DS   F     Maximum segment length
MQCXP_EXITUSERAREA      DS   XL16  Exit user area
MQCXP_EXITDATA          DS   CL32  Exit data
MQCXP_MSGRETRYCOUNT     DS   F     Number of times the message has been
*                                  retried
MQCXP_MSGRETRYINTERVAL  DS   F     Minimum interval in milliseconds
*                                  after which the put operation should
*                                  be retried
MQCXP_MSGRETRYREASON    DS   F     Reason code from previous attempt to
*                                  put the message
MQCXP_HEADERLENGTH      DS   F     Length of header information
MQCXP_PARTNERNAME       DS   CL48  Partner Name
MQCXP_FAPLEVEL          DS   F     Negotiated Formats and Protocols
*                                  level
MQCXP_CAPABILITYFLAGS   DS   F     Capability flags
MQCXP_EXITNUMBER        DS   F     Exit number
MQCXP_EXITSPACE         DS   F     Number of bytes in transmission
*                                  buffer reserved for exit to use
MQCXP_SSLCERTUSERID     DS   CL12  User identifier associated with
*                                  remote SSL certificate
MQCXP_SSLREMCERTISSNAMELENGTH   DS   F     Length of distinguished name
*                                  of issuer of remote SSL certificate
MQCXP_SSLREMCERTISSNAMEPTR      DS   F     Address of distinguished name
*                                  of issuer of remote SSL certificate
MQCXP_SECURITYPARMS     DS   F    Address of security parameters
MQCXP_CURHDRCOMPRESSION         DS   F     Header data compression used for
*                                  current message
MQCXP_CURMSGCOMPRESSION         DS   F      Message data compression used for
*                                  current message
MQCXP_HCONN             DS   F Connection handle
MQCXP_SHARINGCONVERSATIONS      DS   F Multiple conversations possible on
*                                  channel inst?
MQCXP_MCAUSERSOURCE     DS   F    Source of the provided MCA user ID

MQCXP_LENGTH            EQU  *-MQCXP
                        ORG  MQCXP
MQCXP_AREA              DS   CL(MQCXP_LENGTH)
```

MQXWD - Exit-Wait-Deskriptor

Die MQXWD-Struktur ist ein Ein-/Ausgabeparameter im MQXWAIT-Aufruf.

Diese Struktur wird nur unter z/OS unterstützt.

Zugehörige Verweise

„Felder” auf Seite 1114
In diesem Kapitel sind die in der MQXWD-Struktur enthaltenen Felder aufgeführt und beschrieben.

„Deklaration in Programmiersprache C” auf Seite 1115
Deklaration der MQXWD-Struktur in C

„Deklaration in System/390 Assembler” auf Seite 1115
Deklaration der MQXWD-Struktur in System/390-Assembler

Felder

In diesem Kapitel sind die in der MQXWD-Struktur enthaltenen Felder aufgeführt und beschrieben.

StrucId (MQCHAR4)
Dieses Feld gibt die Struktur-ID an.

Folgende Werte sind möglich:

MQXWD_STRUC_ID
ID für die Exit-Wartedeskriptor-Struktur

Für die Programmiersprache C ist auch die Konstante MQXWD_STRUC_ID_ARRAY definiert. Diese Konstante hat den gleichen Wert wie die Konstante MQXWD_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Der Anfangswert dieses Felds ist MQXWD_STRUC_ID.

Version (MQLONG)
Dieses Feld gibt die Strukturversionsnummer an.

Folgende Werte sind möglich:

MQXWD_VERSION_1
Versionsnummer der Exit-Wartedeskriptor-Struktur

Der Anfangswert dieses Felds ist MQXWD_VERSION_1.

Reserved1 (MQLONG)
Dieses Feld ist reserviert. Der Wert muss null sein.

Dies ist ein Eingabefeld.

Reserved2 (MQLONG)
Dieses Feld ist reserviert. Der Wert muss null sein.

Dies ist ein Eingabefeld.

Reserved3 (MQLONG)
Dieses Feld ist reserviert. Der Wert muss null sein.

Dies ist ein Eingabefeld.

ECB (MQLONG)
Ereignissteuerblock mit Wartestatus

Dieses Feld gibt den Ereignissteuerblock (ECB) mit Wartestatus an.Es muss auf null gesetzt werden, bevor der MQXWAIT-Aufruf ausgegeben wird. Nach erfolgreicher Beendigung enthält es den Bereitstellungscode.

Dies ist ein Ein-/Ausgabefeld.

Deklaration in Programmiersprache C
Deklaration der MQXWD-Struktur in C

```
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 */
};
```

Deklaration in System/390 Assembler
Deklaration der MQXWD-Struktur in System/390-Assembler

```
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)
```

API-Exitreferenz

Dieser Abschnitt enthält Referenzinformationen, die in erster Linie für einen Programmierer, der API-Exits schreibt, von Interesse sind.

Allgemeine Hinweise zur Verwendung

Hinweise:

1. Alle Exitfunktionen können den MQXEP-Aufruf ausgeben; dieser Aufruf ist speziell für die Verwendung durch API-Exitfunktionen gedacht.
2. Die Funktion MQ_INIT_EXIT kann keine anderen MQ-Aufrufe als MQXEP ausgeben.
3. Sie können den Aufruf MQDISC nicht für die aktuelle Verbindung absetzen.
4. Wenn eine Exitfunktion den MQCONN-Aufruf oder den MQCONNX-Aufruf mit der Option MQCNO_HANDLE_SHARE_NONE ausgibt, wird der Aufruf mit dem Ursachencode MQRC_ALREADY_CONNECTED abgeschlossen, und die zurückgegebene Kennung ist mit der an den Exit als Parameter übergebenen Kennung identisch.
5. Wenn eine API-Exitfunktion einen MQI-Aufruf ausgibt, werden API-Exits im Allgemeinen nicht rekursiv aufgerufen. Wenn jedoch eine Exitfunktion den MQCONNX-Aufruf mit der Option MQCNO_HANDLE_SHARE_BLOCK oder MQCNO_HANDLE_SHARE_NO_BLOCK ausgibt, gibt der Aufruf eine neue gemeinsam genutzte Kennung zurück. Dadurch erhält die Exit-Suite eine eigene Verbindungskennung, also eine Arbeitseinheit, die unabhängig von der Arbeitseinheit der Anwendung ist. Die Exit-Suite kann mithilfe dieser Kennung Nachrichten innerhalb ihrer eigenen Arbeitseinheit einreihen und abrufen und diese Arbeitseinheit festschreiben oder zurücksetzen. All dies lässt sich ohne Beeinträchtigung der Arbeitseinheit der Anwendung durchführen.

   Da die Exitfunktion eine andere Verbindungskennung als die Anwendung verwendet, bewirken MQ-Aufrufe, die von der Exitfunktion abgesetzt werden, dass die relevanten API-Exitfunktionen aufgerufen werden. Daher können Exitfunktionen rekursiv aufgerufen werden. Beachten Sie, dass sowohl das Feld *ExitUserArea* in MQAXP als auch der Bereich der Exitkette den Geltungsbereich der Verbindungskennung haben. Daher kann eine Exitfunktion diese Bereiche nicht dazu verwenden, um einer anderen, rekursiv aufgerufenen Instanz von sich selbst zu signalisieren, dass sie bereits aktiv ist.
6. Exitfunktionen können auch Nachrichten innerhalb der Arbeitseinheit der Anwendung einreihen und abrufen. Wenn die Anwendung die Arbeitseinheit festschreibt oder zurücksetzt, werden alle Nachrichten in der Arbeitseinheit gemeinsam festgeschrieben oder zurückgesetzt, unabhängig davon, wer sie in der Arbeitseinheit (Anwendung oder Exitfunktion) platziert hat. Der Exit kann jedoch bewirken, dass die Anwendung die Systemgrenzen früher überschreitet als sonst (indem beispielsweise die maximale Anzahl der nicht festgeschriebenen Nachrichten in einer Arbeitseinheit überschritten wird).

   Wenn eine Exitfunktion die Arbeitseinheit der Anwendung auf diese Weise verwendet, sollte die Exitfunktion normalerweise vermeiden, den MQCMIT-Aufruf auszugeben, da dies die Arbeitseinheit der Anwendung festschreibt und möglicherweise das korrekte Funktionieren der Anwendung beeinträchtigt. Allerdings muss die Exitfunktion möglicherweise in manchen Fällen den MQBACK-Aufruf ausgeben, wenn die Exitfunktion einen schweren Fehler feststellt, der die Festschreibung der Arbeitseinheit verhindert (beispielsweise ein Fehler beim Einreihen einer Nachricht als Teil der Arbeitseinheit der Anwendung). Stellen Sie beim Aufruf von MQBACK unbedingt sicher, dass die Begrenzungen der Anwendungsarbeitseinheit nicht geändert werden. In dieser Situation muss die Exitfunktion die entsprechenden Werte festlegen, um sicherzustellen, dass der Beendigungscode MQCC_WARNING

und der Ursachencode MQRC_BACKED_OUT an die Anwendung zurückgegeben werden, damit die Anwendung erkennen kann, dass die Arbeitseinheit zurückgesetzt wurde.

Wenn eine Exitfunktion mithilfe der Verbindungskennung der Anwendung MQ-Aufrufe absetzt, führen diese Aufrufe selbst nicht zu weiteren Aufrufen von API-Exitfunktionen.

7. Wenn eine MQXR_BEFORE-Exitfunktion fehlerhaft beendet wird, kann der Warteschlangenmanager möglicherweise von dem Fehler wiederhergestellt werden. In diesem Fall fährt der Warteschlangenmanager mit der Verarbeitung so fort, als hätte die Exitfunktion MQXCC_FAILED zurückgegeben. Wenn der Warteschlangenmanager nicht wiederhergestellt werden kann, wird die Anwendung beendet.
8. Wenn eine MQXR_AFTER-Exitfunktion fehlerhaft beendet wird, kann der Warteschlangenmanager möglicherweise von dem Fehler wiederhergestellt werden. In diesem Fall fährt der Warteschlangenmanager mit der Verarbeitung so fort, als hätte die Exitfunktion MQXCC_FAILED zurückgegeben. Wenn der Warteschlangenmanager nicht wiederhergestellt werden kann, wird die Anwendung beendet. Denken Sie daran, dass im letzten Fall die außerhalb einer Arbeitseinheit abgerufenen Nachrichten verloren gehen (dies entspricht der Situation, wenn die Anwendung sofort nach dem Entfernen einer Nachricht aus der Warteschlange fehlschlägt).
9. Der MCA-Prozess führt eine zweiphasige Festschreibung aus.

   Wenn ein API-Exit einen MQCMIT aus einem vorbereiteten MCA-Prozess abfängt und versucht, eine Aktion innerhalb der Arbeitseinheit auszuführen, schlägt die Aktion mit dem Ursachencode MQRC_UOW_NOT_AVAILABLE fehl.
10. In einer Umgebung mit verschiedenen Installationen funktioniert ein Exit nur dann mit WebSphere MQ Version 7.0 und Version 7.1, wenn der Exit so geschrieben ist, dass er sich in Version 7.0 mit mqm.Lib verknüpft. Bei nicht primären bzw. verlagerten Exits muss zudem sichergestellt werden, dass die Anwendung, bevor sie gestartet wird, die mqm.Lib der Installation findet, mit der der Warteschlangenmanager zum jeweiligen Zeitpunkt verknüpft ist. (Hierfür kann vor dem Start der Anwendung zum Beispiel der Befehl **setmqenv -m QM** ausgeführt werden, selbst wenn der Warteschlangenmanager zu einer Installation der Version 7.0 gehört.)
11. Bei verschiedenen Installationen von IBM WebSphere MQ sollten Sie die für die älteste Version von IBM WebSphere MQ entwickelten Exits verwenden, da die in späteren Versionen hinzugekommenen Funktionen in den Vorgängerversionen vermutlich nicht funktionieren. Weitere Informationen zu den Änderungen, die in der neuen Version vorgenommen wurden, finden Sie im Abschnitt Neuerungen in WebSphere MQ 7.5.

API-Exitparameterstruktur von IBM WebSphere MQ (MQAXP)

Die MQAXP-Struktur, ein externer Steuerblock, wird als ein Eingabe- oder Ausgabeparameter zum API-Exit verwendet. Dieser Abschnitt enthält außerdem Informationen darüber, wie Warteschlangenmanager Exitfunktionen verarbeiten.

MQAXP hat folgende Deklaration in 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 */
};
```

Die folgende Parameterliste wird übergeben, wenn Funktionen in einem API-Exit aufgerufen werden:

StrucId (MQCHAR4) - Eingabe
Die Exitparameter-Struktur-ID mit einem Wert von:

```
MQAXP_STRUC_ID.
```

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

Version (MQLONG) - Eingabe
Die Strukturversionsnummer mit einem Wert von:

MQAXP_VERSION_1
Version 1 API-Exitparameterstruktur.

MQAXP_VERSION_2
Version 2 API-Exitparameterstruktur.

MQAXP_CURRENT_VERSION
Aktuelle Versionsnummer für die API-Exitparameterstruktur.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

ExitId (MQLONG) - Eingabe
Die Exit-ID mit der Einstellung auf Eingabe zur Exitroutine gibt den Exittyp an:

MQXT_API_EXIT
API-Exit.

ExitReason (MQLONG) - Eingabe
Die Ursache für den Aufruf des Exits, auf Eingabe zu jeder Exitfunktion eingestellt:

MQXR_CONNECTION
Der Exit wird aufgerufen, um sich selbst zu initialisieren, bevor ein MQCONN- oder MQCONNX-Aufruf erfolgt, oder selbst vor einem MQDISC-Aufruf zu enden.

MQXR_BEFORE
Der Exit wird abgerufen, bevor er einen API-Aufruf ausführt oder bevor er Daten auf einem MQGET konvertiert.

MQXR_AFTER
Der Exit wird nach Ausführen eines API-Aufrufs aufgerufen.

ExitResponse (MQLONG) - Ausgabe
Die Antwort vom Exit, initialisiert bei Eingabe in jede Exitfunktion zu:

MQXCC_OK
Normal fortsetzen.

Dieses Feld muss von der Exitfunktion eingestellt werden, um dem Warteschlangenmanager das Ergebnis der ausgeführten Exitfunktion mitzuteilen. Folgende Werte sind möglich:

MQXCC_OK
Die Exitfunktion wurde erfolgreich abgeschlossen. Normal fortsetzen.

Dieser Wert kann von allen MQXR_*-Exitfunktionen eingestellt werden. ExitResponse2 wird verwendet, um zu entscheiden, ob Exitfunktionen später in der Kette aufgerufen werden.

MQXCC_FAILED
Die Exitfunktion ist aufgrund eines Fehlers fehlgeschlagen.

Dieser Wert kann von allen MQXR_*-Exitfunktionen eingestellt werden. Der Warteschlangenmanager stellt CompCode auf MQCC_FAILED und Reason auf:

- MQRC_API_EXIT_INIT_ERROR, wenn die Funktion MQ_INIT_EXIT ist
- MQRC_API_EXIT_TERM_ERROR, wenn die Funktion MQ_TERM_EXIT ist

- MeQRC_API_EXIT_ERROR für alle anderen Exitfunktionen

Die eingestellten Werte können durch eine Exitfunktion später in der Kette geändert werden.

ExitResponse2 wird ignoriert; der Warteschlangenmanager fährt mit der Verarbeitung fort, als wäre MQXR2_SUPPRESS_CHAIN zurückgegeben worden.

MQXCC_SUPPRESS_FUNCTION
Unterdrücken der WebSphere MQ API-Funktion.

Dieser Wert kann nur durch eine MQXR_BEFORE-Exitfunktion eingestellt werden. Der API-Aufruf wird umgangen. Wenn er vom MQ_DATA_CONV_ON_GET_EXIT zurückgegeben wird, wird die Datenkonvertierung umgangen. Der Warteschlangenmanager stellt CompCode auf MQCC_FAILED und Reason auf MQRC_SUPPRESSED_BY_EXIT, aber die eingestellten Werte können durch eine Exitfunktion später in der Kette geändert werden. Andere Parameter für den Aufruf bleiben so, wie sie beim Verlassen des Exits waren. ExitResponse2 wird verwendet, um zu entscheiden, ob Exitfunktionen später in der Kette aufgerufen werden.

Wenn dieser Wert von einer MQXR_AFTER- oder MQXR_CONNECTION-Exitfunktion eingestellt wird, setzt der Warteschlangenmanager die Verarbeitung fort, als ob MQXCC_FAILED zurückgegeben worden wäre.

MQXCC_SKIP_FUNCTION
Überspringen der WebSphere MQ API-Funktion.

Dieser Wert kann nur durch eine MQXR_BEFORE-Exitfunktion eingestellt werden. Der API-Aufruf wird umgangen. Wenn er vom MQ_DATA_CONV_ON_GET_EXIT zurückgegeben wird, wird die Datenkonvertierung umgangen. Die Exitfunktion muss bei den Werten CompCode und Reason einstellen, um zur Anwendung zurückgegeben zu werden, aber die eingestellten Werte können durch eine Exitfunktion später in der Kette geändert werden. Andere Parameter für den Aufruf bleiben so, wie sie beim Verlassen des Exits waren. ExitResponse2 wird verwendet, um zu entscheiden, ob Exitfunktionen später in der Kette aufgerufen werden.

Wenn dieser Wert von einer MQXR_AFTER- oder MQXR_CONNECTION-Exitfunktion eingestellt wird, setzt der Warteschlangenmanager die Verarbeitung fort, als ob MQXCC_FAILED zurückgegeben worden wäre.

MQXCC_SUPPRESS_EXIT
Unterdrücken aller Exitfunktionen, die zur Einstellung der Exits gehören.

Dieser Wert kann nur durch die Exitfunktionen MQXR_BEFORE und MQXR_AFTER eingestellt werden. Er umgeht *alle* nachfolgenden Aufrufe der Exitfunktionen, die zu dieser Einstellung von Exits für diese logische Verbindung gehören. Diese Umgehung wird solange fortgesetzt, bis die logische Unterbrechungsanforderung auftritt, wenn die MQ_TERM_EXIT-Funktion mit ExitReason von MQXR_CONNECTION aufgerufen wird.

Die Exitfunktion muss bei den Werten CompCode und Reason einstellen, um zur Anwendung zurückgegeben zu werden, aber die eingestellten Werte können durch eine Exitfunktion später in der Kette geändert werden. Andere Parameter für den Aufruf bleiben so, wie sie beim Verlassen des Exits waren. ExitResponse2 wird ignoriert.

Wenn dieser Wert von einer MQXR_CONNECTION-Exitfunktion eingestellt wird, setzt der Warteschlangenmanager die Verarbeitung fort, als ob MQXCC_FAILED zurückgegeben worden wäre.

Weitere Informationen über die Interaktion zwischen ExitResponse und ExitResponse2 sowie über deren Auswirkung auf die Exitverarbeitung finden Sie unter „So verarbeiten Warteschlangenmanager Exitfunktionen” auf Seite 1122.

ExitResponse2 (MQLONG) - Ausgabe
Das ist ein sekundärer Exitantwortcode, der den primären Exitantwortcode für MQXR_BEFORE-Exitfunktionen qualifiziert. Er wird für Folgendes initialisiert:

```
MQXR2_DEFAULT_CONTINUATION
```

bei Eingabe in eine WebSphere MQ API-Aufruf-Exitfunktion. Er kann auf einen der folgenden Werte eingestellt werden:

MQXR2_DEFAULT_CONTINUATION
Ob mit dem nächsten Exit in der Kette fortgefahren wird, je nach dem Wert von ExitResponse.

Wenn ExitResponse MQXCC_SUPPRESS_FUNCTION oder MQXCC_SKIP_FUNCTION ist, Umgehung von Exitfunktionen später in der MQXR_BEFORE-Kette und der übereinstimmenden Exitfunktionen in der MQXR_AFTER-Kette. Aufrufen von Exitfunktionen in der MQXR_AFTER-Kette, die mit Exitfunktionen vorne in der MQXR_BEFORE-Kette übereinstimmen.

Andernfalls wird der nächste Exit in der Kette aufgerufen.

MQXR2_SUPPRESS_CHAIN
Unterdrücken der Kette.

Umgehung von Exitfunktionen später in der MQXR_BEFORE-Kette und der übereinstimmenden Exitfunktionen in der MQXR_AFTER-Kette für diesen API-Aufruf. Aufrufen von Exitfunktionen in der MQXR_AFTER-Kette, die mit Exitfunktionen vorne in der MQXR_BEFORE-Kette übereinstimmen.

MQXR2_CONTINUE_CHAIN
Fortfahren mit dem nächsten Exit in der Kette.

Weitere Informationen über die Interaktion zwischen ExitResponse und ExitResponse2 sowie über deren Auswirkung auf die Exitverarbeitung finden Sie unter „So verarbeiten Warteschlangenmanager Exitfunktionen” auf Seite 1122.

Feedback (MQLONG) - Ein-/Ausgabe
Kommunizieren von Rückmeldungscodes zwischen Exitfunktionsaufrufen. Das wird aus folgendem Grund initialisiert:

```
MQFB_NONE (0)
```

vor der ersten Funktion des ersten Exits in einer Kette.

Exits können bei diesem Feld jeden Wert einstellen, einschließlich aller gültigen MQFB_*- oder MQRC_*-Werte. Exits können bei diesem Feld außerdem einen benutzerdefinierten Rückmeldungswert im Bereich von MQFB_APPL_FIRST bis MQFB_APPL_LAST einstellen.

APICallerType (MQLONG) - Eingabe
Der API-Aufrufertyp, der angibt, ob der WebSphere MQ API-Aufrufer innerhalb oder außerhalb des Warteschlangenmanagers ist: MQXACT_EXTERNAL oder MQXACT_INTERNAL.

ExitUserArea (MQBYTE16) - Ein-/Ausgabe
Ein Benutzerbereich, der für alle Exits mit einem besonderen ExitInfoObject verfügbar ist. Er wird für MQXUA_NONE (binäre Nullen für die Länge des ExitUserArea) initialisiert, bevor die erste Exitfunktion (MQ_INIT_EXIT) für den hconn aufgerufen wird. Ab diesem Punkt werden alle Änderungen, die in diesem Feld durch Exitfunktion erfolgen, über Aufrufe desselben Exits hinweg erhalten.

Dieses Feld wird auf ein Vielfaches von 4 MQLONGs ausgerichtet.

Exits können außerdem zu jedem Speicher eine Anbindung herstellen, die sie von diesem Bereich aus anlegen.

Für jedes hconn hat jeder Exit in einer Kette von Exits einen unterschiedlichen ExitUserArea. Der ExitUserArea kann nicht von Exits in einer Kette gemeinsam genutzt werden und die Inhalte des ExitUserArea für einen Exit sind für einen anderen Exit in der Kette nicht verfügbar.

Bei Programmen in C wird die Konstante MQXUA_NONE_ARRAY außerdem mit demselben Wert wie MQXUA_NONE definiert, aber als eine Feldgruppe aus Zeichen anstatt aus Zeichenfolgen.

Die Länge dieses Felds wird durch MQ_EXIT_USER_AREA_LENGTH angegeben.

ExitData (MQCHAR32) - Eingabe
Exitdaten, die bei Eingabe für jede Exitfunktion der 32 Zeichen der exit-spezifischen Daten eingestellt werden, die im Exit bereitgestellt werden. Wenn Sie keinen Wert im Exit definieren, ist dieses Feld ganz leer.

Die Länge dieses Felds ist durch MQ_EXIT_DATA_LENGTH vorgegeben.

ExitInfoName (MQCHAR48) - Eingabe
Der Exitinformationsname, der bei Eingabe für jede Exitfunktion für den ApiExit_name eingestellt wird, der in den Exitdefinitionen in den Zeilengruppen angegeben wird.

ExitPDArea (MQBYTE48) - Ein-/Ausgabe
Ein Problembestimmungbereich, der für MQXPDA_NONE (binäre Nullen für die Länge des Felds) für jeden Aufruf einer Exitfunktion initialisiert wird.

Bei Programmen in C wird die Konstante MQXPDA_NONE_ARRAY außerdem mit demselben Wert wie MQXPDA_NONE definiert, aber als eine Feldgruppe aus Zeichen anstatt aus Zeichenfolgen.

Die Exitverwaltung schreibt diesen Bereich immer in den WebSphere MQ-Trace am Ende des Exits, auch wenn die Funktion erfolgreich ist.

Die Länge dieses Felds wird durch MQ_EXIT_PD_AREA_LENGTH vorgegeben.

QMgrName (MQCHAR48) - Eingabe
Der Name des Warteschlangenmanagers, mit dem die Anwendung verbunden ist, der in Folge der Verarbeitung eines WebSphere MQ API-Aufrufs einen Exit aufgerufen hat.

Wenn der Name des Warteschlangenmanagers, der bei MQCONN- oder MQCONNX-Aufrufen bereitgestellt wird, leer ist, wird dieses Feld dennoch auf den Namen des Warteschlangenmanagers eingestellt, mit dem die Anwendung verbunden ist, egal ob die Anwendung der Server oder der Client ist.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

Die Länge dieses Feldes wird durch MQ_Q_MGR_NAME_LENGTH angegeben.

ExitChainAreaPtr (PMQACH) - Ein-/Ausgabe
Dies wird verwendet, um Daten über Aufrufe verschiedener Exits in einer Kette zu übertragen. Es wird auf einen NULL-Zeiger eingestellt, bevor die erste Funktion (MQ_INIT_EXIT mit ExitReason MQXR_CONNECTION) des ersten Exits in einer Kette von Exits aufgerufen wird. Der vom Exit bei Aufruf zurückgegebene Wert wird zum nächsten Aufruf weitergegeben.

Weitere Informationen zur Verwendung des Exitkettenbereichs finden Sie unter „Der Exitkettenbereich und Exitkettenbereichsheader (MQACH)” auf Seite 1125.

Hconfig (MQHCONFIG) - Eingabe
Die Konfigurationskennung, die die Einstellung der Funktionen darstellt, die initialisiert werden. Dieser Wert wird vom Warteschlangenmanager in der MQ_INIT_EXIT-Funktion erstellt und wird später an die API-Exitfunktion übergeben. Er wird bei Eingabe in jede Exitfunktion eingestellt.

Sie können Hconfig als einen Zeiger zur MQIEP-Struktur verwenden, um MQI- und DCI-Aufruf zu starten. Sie müssen prüfen, ob die ersten 4 Bytes von HConfig mit der StrucId der MQIEP-Struktur übereinstimmen, bevor Sie die HConfig-Parameter als Zeiger zur MQIEP-Struktur verwenden.

Function (MQLONG) - Eingabe
Die Funktions-ID, gültige Werte, für die die MQXF_*-Konstanten in „Externe Konstanten” auf Seite 1127 beschrieben werden.

Die Exitverwaltung stellt dieses Feld bei Eingabe in jede Exitfunktion auf den korrekten Wert ein, je nach dem WebSphere MQ API-Aufruf, der aus dem aufgerufenen Exit resultiert.

ExitMsgHandle (MQHMSG) - Ein-/Ausgabe
Wenn Function MQXF_GET und ExitReason MQXR_AFTER ist, wird eine gültige Nachrichtenkennung in diesem Feld zurückgegeben, die dem API-Exit den Zugriff auf die Nachrichtendeskriptorfelder und alle anderen Eigenschaften ermöglicht, die mit der ExitProperties-Zeichenfolge in der MQXEPO-Struktur übereinstimmen, wenn der API-Exit registriert wird.

Alle Nicht-Nachrichtendeskriptoreigenschaften, die in der ExitMsgHandle zurückgegeben werden, werden von der MsgHandle in der MQGMO-Struktur (wenn eine angegeben wurde) oder in den Nachrichtendaten nicht verfügbar sein.

Wenn Function MQXF_GET und ExitReason MQXR_BEFORE ist, wenn das Exitprogramm dieses Feld auf MQHM_NONE einstellt, dann wird das Auffüllen der ExitMsgHandle-Eigenschaften unterdrückt.

Dieses Feld wird nicht eingestellt, wenn Version kleiner als MQAXP_VERSION_2 ist.

So verarbeiten Warteschlangenmanager Exitfunktionen

Die Verarbeitung, die der Warteschlangenmanager von einer Exitfunktion aus bei der Rückgabe ausführt, hängt von ExitResponse sowie ExitResponse2 ab.

In Tabelle 593 auf Seite 1122 werden die möglichen Kombinationen und deren Auswirkung auf eine MQXR_BEFORE-Exitfunktion zusammengefasst und es wird dargestellt:

- Wer die Parameter CompCode und Reason des API-Aufrufs einstellt
- Ob die verbleibenden Exitfunktionen in der MQXR_BEFORE-Kette und die übereinstimmenden Exitfunktionen in der MQXR_AFTER-Kette aufgerufen werden
- Ob der API-Aufruf aufgerufen wird

Für eine MQXR_AFTER-Exitfunktion:

- CompCode und Reason werden in gleicher Art wie MQXR_BEFORE eingestellt
- ExitResponse2 wird ignoriert (die in der MQXR_AFTER-Kette verbleibenden Exitfunktionen werden aufgerufen)
- MQXCC_SUPPRESS_FUNCTION und MQXCC_SKIP_FUNCTION sind nicht gültig

Für eine MQXR_CONNECTION-Exitfunktion:

- CompCode und Reason werden in gleicher Art wie MQXR_BEFORE eingestellt
- ExitResponse2 wird ignoriert.
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT sind nicht gültig

In allen Fällen, in denen ein Exit oder der Warteschlangenmanager CompCode und Reason einstellt, können die eingestellten Werte später durch einen aufgerufenen Exit oder durch den API-Aufruf geändert werden (wenn der API-Aufruf später aufgerufen wird).

Tabelle 593. MQXR_BEFORE Exitverarbeitung

Wert von ExitResponse	CompCode und Reason eingestellt von	Wert von ExitResponse2 (Standardfortsetzung) Chain	Wert von ExitResponse2 (Standardfortsetzung) API
MQXCC_OK	exit	Y	Y
MQXCC_SUPPRESS_EXIT	exit	Y	Y
MQXCC_SUPPRESS_FUNCTION	Warteschlangenmanager	N	N
MQXCC_SKIP FUNCTION	exit	N	N
MQXCC_FAILED	Warteschlangenmanager	N	N

So verarbeiten Clients Exitfunktionen

Generell verarbeiten Clients Exitfunktionen in der gleichen Art wie Serveranwendungen und das *QMgrName*-Attribut in dieser Struktur gilt, egal ob sich die Funktion auf einem Server oder einem Client befindet.

Der Client hat jedoch keine Vorstellung von der *mqs.ini*-Datei, sodass die Zeilengruppen *ApiExitCommon* und *APIExitTemplate* nicht gelten. Es gilt nur die Zeilengruppe *ApiExitLocal*, und diese Zeilengruppe wird in der *mqclient.ini*-Datei konfiguriert.

IBM WebSphere MQ-API-Exit-Kontextstruktur (MQAXC)

Die MQAXC-Struktur, ein externer Steuerblock, wird als Eingabeparameter bei einem API-Exit verwendet.

MQAXC verfügt über die folgende Deklaration für 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 */
 };
```

Die Parameter zu MQAXC sind:

StrucId (MQCHAR4) - Eingabe
Die Exitkontextstruktur-ID, mit einem Wert von MQAXC_STRUC_ID. Bei Programmen in C wird die MQAXC_STRUC_ID_ARRAY ebenfalls mit demselben Wert wie MQAXC_STRUC_ID definiert, aber als eine Feldgruppe von Zeichen anstatt aus Zeichenfolgen.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

Version (MQLONG) - Eingabe
Die Strukturversionsnummer mit einem Wert von:

MQAXC_VERSION_2
Versionsnummer für die Exitkontextstruktur.

MQAXC_CURRENT_VERSION
Aktuelle Versionsnummer für die Exitkontextstruktur.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

Environment (MQLONG) - Eingabe
Die Umgebung, von der ein WebSphere MQ API-Aufruf ausgegeben wurde, der dazu führte, dass eine Exitfunktion ausgeführt wurde. Gültige Werte für dieses Feld sind:

MQXE_OTHER
Dieser Wert ist konsistent mit Aufrufen, die ein API-Exit sieht, wenn der Exit von einer Serveranwendung aufgerufen wird. Das bedeutet, dass ein API-Exit unverändert auf einem Client ausgeführt wird keine Veränderung wahrnimmt.

Wenn der Exit tatsächlich festlegen muss, ob er auf dem Client ausgeführt wird, kann der Exit dies tun, indem er auf die *ChannelName*- und *ChannelDefinition*-Felder schaut.

MQXE_MCA
Nachrichtenkanalagent

MQXE_MCA_SVRCONN
Ein Nachrichtenkanalagent, der stellvertretend für einen Client agiert

MQXE_COMMAND_SERVER
Befehlsserver

MQXE_MQSC
Der Befehlsinterpreter runmqsc

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

UserId (MQCHAR12) - Eingabe
Die Benutzer-ID, die der Anwendung zugeordnet ist. Besonders im Fall von Clientverbindungen enthält dieses Feld im Gegensatz zur Benutzer-ID, unter der der Kanalcode ausgeführt wird, die Benutzer-ID vom angenommenen Benutzer. Wenn eine leere Benutzer-ID vom Client übertragen wird, erfolgt bei der bereits verwendeten Benutzer-ID keine Änderung. Das heißt, dass keine neue Benutzer-ID angenommen wird.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe. Die Länge dieses Felds wird durch MQ_USER_ID_LENGTH angegeben.

Im Falle eines Clients ist dies die Benutzer-ID, die vom Client an den Server gesendet wird. Beachten Sie, dass dies möglicherweise nicht die aktuelle Benutzer-ID ist, auf der der Client im Warteschlangenmanager ausgeführt wird, da es eine MCAUser- oder CHLAUTH-Konfiguration geben könnte, die die Benutzer-ID ändert.

SecurityId (MQBYTE40) - Eingabe
Eine Erweiterung der Benutzer-ID, die die Anwendung ausführt. Ihre Länge wird durch MQ_SECURITY_ID_LENGTH angegeben.

Im Falle eines Clients ist dies die Benutzer-ID, die vom Client an den Server gesendet wird. Beachten Sie, dass dies möglicherweise nicht die aktuelle Benutzer-ID ist, auf der der Client im Warteschlangenmanager ausgeführt wird, da es eine MCAUser- oder CHLAUTH-Konfiguration geben könnte, die die Benutzer-ID ändert.

ConnectionName (MQCHAR264) - Eingabe
Das Verbindungsnamensfeld, das auf die Adresse des Clients eingestellt ist. Bei TCP/IP wäre es z. B. die Client-IP-Adresse.

Die Länge dieses Feldes ist durch MQ_CONN_NAME_LENGTH vorgegeben.

Im Falle eines Clients ist dies die Partneradresse des Warteschlangenmanagers.

LongMCAUserIdLength (MQLONG) - Eingabe
Die Länge der langen Nachrichtenkanalagent-Benutzer-ID.

Wenn ein Nachrichtenkanalagent eine Verbindung mit dem Warteschlangenmanager herstellt, wird dieses Feld auf die Länge der langen Nachrichtenkanalagent-Benutzer-ID eingestellt (oder Null, wenn es keine derartige ID gibt).

Im Falle eines Clients ist dies die Client-long-Benutzer-ID.

LongRemoteUserIdLength (MQLONG) - Eingabe
Die Länge der langen Remotebenutzer-ID.

Wenn der Nachrichtenkanalagent eine Verbindung mit dem Warteschlangenmanager herstellt, wird dieses Feld auf die Länge der langen Remotebenutzer-ID ID eingestellt. Andernfalls wird das Feld auf Null eingestellt

Stellen Sie dieses Feld im Falle eines Clients auf Null.

LongMCAUserIdPtr (MQPTR) - Eingabe
Adresse der langen Nachrichtenkanalagent-Benutzer-ID.

Wenn ein Nachrichtenkanalagent eine Verbindung mit dem Warteschlangenmanager herstellt, wird dieses Feld auf die Adresse der langen Nachrichtenkanalagent-Benutzer-ID eingestellt (oder auf einen Nullzeiger, wenn es keine derartige ID gibt).

Im Falle eines Clients ist dies die Client-long-Benutzer-ID.

LongRemoteUserIdPtr (MQPTR) - Eingabe
Die Adresse der langen Remotebenutzer-ID.

Wenn ein Nachrichtenkanalagent eine Verbindung mit dem Warteschlangenmanager herstellt, wird dieses Feld auf die Adresse der langen Remotebenutzer-ID eingestellt (oder auf einen Nullzeiger, wenn es keine derartige ID gibt).

Stellen Sie dieses Feld im Falle eines Clients auf Null.

ApplName (MQCHAR28) - Eingabe
Der Name der Anwendung oder Komponente, die der WebSphere MQ API-Aufruf ausgibt.

Für die Erstellung des ApplName gelten dieselben Regeln wie für die Erstellung des Standardnamens für ein MQPUT.

Der Wert dieses Feldes wird ermittelt, indem der Programmname beim Betriebssystem abgefragt wird. Seine Länge wird durch MQ_APPL_NAME_LENGTH angegeben.

ApplType (MQLONG) - Eingabe
Der Typ der Anwendung oder Komponente, die der WebSphere MQ API-Aufruf ausgibt.

Für die Plattform, auf der die Anwendung kompiliert wird, ist der Wert MQAT_DEFAULT oder einer der MQAT_*-Werte.

Die Exitverwaltung stellt dieses Feld bei jeder Exitfunktion auf Eingabe.

ProcessId (MQPID) - Eingabe
Die Betriebssystemprozess-ID.

Wenn zutreffend stellt die Exitverwaltung dieses Feld bei jeder Exitfunktion auf Eingabe.

ThreadId (MQTID) - Eingabe
Die MQ-Thread-ID. Das ist dieselbe ID, die im MQ-Trace und FFST-Speicherauszügen verwendet wird, kann sich jedoch von der ID des Betriebssystem-Threads unterscheiden.

Wenn zutreffend stellt die Exitverwaltung dieses Feld bei jeder Exitfunktion auf Eingabe.

ChannelName (MQCHAR) - Eingabe
Der Name des Kanals, aufgefüllt mit Leerzeichen, falls zutreffend und bekannt.

Falls nicht zutreffend wird dieses Feld auf NULL-Zeichen eingestellt.

Reserved1 (MQBYTE4) - Eingabe
Dieses Feld ist reserviert.

ChanneDefinition (PMQCD) - Eingabe
Ein Zeiger zur Kanaldefinition wird verwendet, falls zutreffend und bekannt.

Falls nicht zutreffend wird dieses Feld auf NULL-Zeichen eingestellt.

Beachten Sie, dass der Zeiger nur abgeschlossen wird, wenn die Verbindung einen Verarbeitungsvorgang stellvertretend für einen WebSphere MQ-Kanal ausführt und diese Kanaldefinition gelesen wurde.

Insbesondere wird die Kanaldefinition nicht auf dem Server angegeben, wenn der erste MQCONN-Aufruf für den Kanal erfolgt. Wenn der Zeiger gefüllt wird, muss die Struktur (und alle Unterstrukturen), auf die der Zeiger verweist, außerdem als schreibgeschützt behandelt werden; eine Aktualisierung der Struktur würde zu unvorhersehbaren Ergebnissen führen und wird nicht unterstützt.

Im Falle eines Clients enthalten Felder Werte, die für eine Clientanwendung zutreffend sind. Dies gilt nicht für Felder mit einem für einen Client angegebenen Wert.

Der Exitkettenbereich und Exitkettenbereichsheader (MQACH)

Falls erforderlich, kann eine Exitfunktion Speicher für einen Exitkettenbereich erfassen und den ExitChainAreaPtr in MQAXP so setzen, dass er auf diesen Speicher verweist.

Exits (entweder dieselben oder abweichende Exitfunktionen) können mehrere Exitkettenbereiche übernehmen und sie miteinander verbinden. Exitkettenbereiche dürfen von dieser Liste nur hinzugefügt oder entfernt werden, wenn sie von der Exitverwaltung aufgerufen werden. Dadurch wird sichergestellt, dass es keine Serialisierungsprobleme durch verschiedene Threads gibt, die sie gleichzeitig von der Liste entfernen oder ihr hinzufügen.

Ein Exitkettenbereich muss mit einer MQACH-Headerstruktur beginnen, die Deklaration in C dafür ist:

```
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 */
 };
```

Die Felder im Exitkettenbereichsheader sind:

StrucId (MQCHAR4) - Eingabe
Die Struktur-ID des Exitkettenbereichs mit einem Anfangswert, der durch MQACH_DEFAULT von MQACH_STRUC_ID definiert wird.

Für C-Programme wird die Konstante MQACH_STRUC_ID_ARRAY ebenfalls definiert; das hat denselben Wert wie MQACH_STRUC_ID, allerdings als Feldgruppe aus Zeichen anstatt einer Zeichenfolge.

Version (MQLONG) - Eingabe
Die Strukturversionsnummer wie folgt:

MQACH_VERSION_1
Die Versionsnummer für die Exitparameterstruktur.

MQACH_CURRENT_VERSION
Die aktuelle Versionsnummer für die Exitkontextstruktur.

Der Anfangswert dieses Feldes, definiert durch MQACH_DEFAULT, ist MQACH_CURRENT_VERSION.

Anmerkung: Wenn Sie eine neue Version dieser Struktur einführen, wird das Layout des vorhandenen Teils nicht geändert. Exitfunktionen müssen prüfen, ob die Versionsnummer gleich oder größer als die niedrigste Version ist, die die Felder umfasst, die von der Exitfunktion verwendet werden sollen.

StrucLength (MQLONG) - Eingabe
Die Länge der MQACH-Struktur. Exits können dieses Feld verwenden, um den Start der Exitdaten festzulegen, um es auf die Länge der vom Exit erstellten Struktur einzustellen.

Der Anfangswert dieses Feldes, definiert durch MQACH_DEFAULT, ist MQACH_CURRENT_LENGTH.

ChainAreaLength (MQLONG) - Eingabe
Die Exitkettenbereichslänge, eingestellt auf die Gesamtlänge des aktuellen Exitkettenbereichs, einschließlich des MQACH-Headers.

Der Anfangswert dieses Felds, definiert durch MQACH_DEFAULT, beträgt null.

ExitInfoName (MQCHAR48) - Eingabe
Der Exitinformationsname.

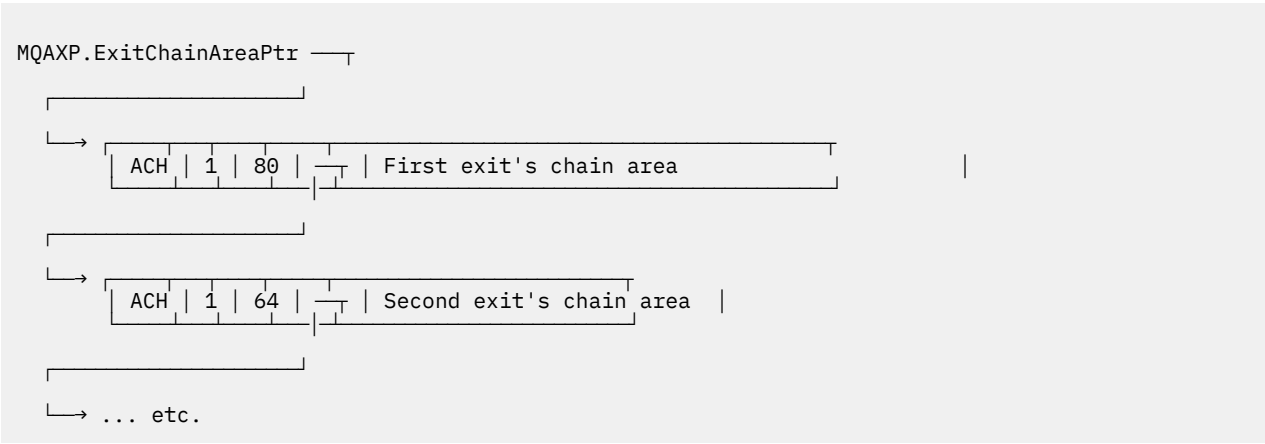
Wenn ein Exit eine MQACH-Struktur erstellt, muss er dieses Feld mit seinem eigenen ExitInfoName initialisieren, sodass diese MQACH-Struktur später entweder von einer anderen Instanz dieses Exits oder von einem mitwirkenden Exit gefunden werden kann.

Der Anfangswert dieses Felds, definiert durch MQACH_DEFAULT, ist eine Zeichenfolge mit Nulllänge ({""}).

NextChainAreaPtr (PMQACH) - Eingabe
Ein Zeiger auf den nächsten Exitkettenbereich mit einem Anfangswert, definiert durch MQACH_DEFAULT, von einem Nullzeiger (NULL).

Exitfunktionen müssen den Speicher für alle für sie erforderlichen Exitkettenbereiche freigeben und die Kettenzeiger ändern, um ihre Exitkettenbereiche von der Liste zu entfernen.

Ein Exitkettenbereich kann wie folgt konstruiert werden:



Externe Konstanten

In diesem Abschnitt erhalten Sie Referenzinformationen für externe Konstanten, die für API-Exits verfügbar sind.

Die folgenden externen Konstanten sind für API-Exits verfügbar:

MQXF_* (Exitfunktion-IDs)

```
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_* (Exit-Ursachen)

```
MQXR_BEFORE                     1       X'00000001'
MQXR_AFTER                      2       X'00000002'
MQXR_CONNECTION                 3       X'00000003'
```

MQXE_* (Umgebungen)

```
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*_* (zusätzliche Konstanten)

```
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*_* (Nullkonstanten)

```
MQXPDA_NONE                     X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY               '\0','\0',...,'\0','\0'
```

MQXCC_* (Beendigungscodes)

```
MQXCC_FAILED                   -8
```

MQRC_* (Ursachencodes)

MQRC_API_EXIT_ERROR 2374 X'00000946'
Ein Exitfunktionsaufruf hat einen ungültigen Antwortcode zurückgegeben oder ist auf andere Weise fehlgeschlagen und der Warteschlangenmanager kann nicht die nächste erforderliche Aktion festlegen.

Prüfen Sie die Felder ExitResponse und ExitResponse2 von MQAXP, um den ungültigen Antwortcode festzulegen, und ändern Sie den Exit, um einen gültigen Antwortcode zurückzugeben.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'
Beim Warteschlangenmanager ist während der Initialisierung der Ausführungsumgebung für eine API-Exitfunktion ein Fehler aufgetreten.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'
Beim Warteschlangenmanager ist beim Schließen der Ausführungsumgebung für eine API-Exitfunktion ein Fehler aufgetreten.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'
Der Wert, der bei einem Registrierungaufruf eines Exiteingangspunkts (MQXEP) für das Feld ExitReason angegeben wurde, ist fehlerhaft.

Prüfen Sie den Wert des ExitReason-Felds, um den ungültigen Exitursachenwert zu bestimmen und korrigieren.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'
Der Wert des Felds Reserved ist fehlerhaft.

Prüfen Sie den Wert des Reserved-Felds, um den Wert Reserved zu bestimmen und korrigieren.

Typedefs für Programmiersprache C

Dieser Abschnitt enthält Informationen über Typedefs im Zusammenhang mit API-Exits, die in der Programmiersprache C verfügbar sind.

Hier finden Sie die Typedefs für Programmiersprache C, die mit den API-Exits in Zusammenhang stehen:

```
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;
```

Der Registrierungsaufruf für den Exiteingangspunkt (MQXEP)

In diesem Abschnitt finden Sie Information über MQXEP, MQXEP-Aufruf in Programmiersprache C und MQXEP-Funktionsprototyp in C.

Verwenden Sie den MQXEP-Aufruf, um:

1. Die Before- und After-Aufrufpunkte von WebSphere MQ API-Exits zu registrieren, bei denen Exitfunktionen aufgerufen werden
2. Die Exitfunktion-Eingangspunkte anzugeben
3. Die Registrierung der Exitfunktion-Eingangspunkte zurückzunehmen

In der Regel werden die MQXEP-Aufrufe in der MQ_INIT_EXIT-Exitfunktion codiert. Sie können sie allerdings in jeder nachfolgenden Exitfunktion angeben.

Wenn Sie einen MQXEP-Aufruf verwenden, um eine bereits registrierte Exitfunktion zu registrieren, wird der zweite MQXEP-Aufruf erfolgreich beendet und ersetzt die registrierte Exitfunktion.

Wenn Sie einen MQXEP-Aufruf verwenden, um eine NULL-Exitfunktion zu registrieren, schließt der MQXEP-Aufruf erfolgreich ab und die Registrierung der Exitfunktion wird zurückgenommen.

Wenn MQXEP-Aufrufe verwendet werden, um eine bestimmte Exitfunktion während der Lebensdauer einer Verbindungsanforderung zu registrieren, um die Registrierung zurückzunehmen oder um sie erneut zu registrieren, wird die vorher registrierte Exitfunktion reaktiviert. Alle Speicher, die dieser Exitfunktionsinstanz noch zugeordnet oder zugehörig sind, sind für die Verwendung durch die Funktionen des Exits verfügbar. (Dieser Speicher wird in der Regel während des Aufrufs der Abschlussexitfunktion freigegeben.)

Die Schnittstelle zu MQXEP ist:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

Dabei gilt:

Hconfig (MQHCONFIG) - Eingabe
Die Konfigurationskennung, die den API-Exit darstellt, zu dem die Funktionen gehören, die initialisiert werden. Dieser Wert wird vom Warteschlangenmanager vor Aufrufen der MQ_INIT_EXIT-Funktion erstellt und wird später an jede API-Exitfunktion im MQAXP übergeben.

ExitReason (MQLONG) - Eingabe
Die Ursache, für die der Eingangspunkt registriert ist, wegen folgender Ursachen:

- Initialisierung oder Beendigung auf Verbindungsebene (MQXR_CONNECTION)
- Vor einem WebSphere MQ API-Aufruf (MQXR_BEFORE)
- Nach einem WebSphere MQ API-Aufruf (MQXR_AFTER)

Function (MQLONG) - Eingabe
Die Funktions-ID, gültige Werte, für die die MQXF_*-Konstanten gelten (siehe „Externe Konstanten" auf Seite 1127).

EntryPoint (PMQFUNC) - Eingabe
Die Adresse des Eingangspunkts für die zu registrierende Exitfunktion. Der Wert NULL gibt entweder an, dass die Exitfunktion nicht bereitgestellt wurde oder dass eine vorherige Registrierung der Exitfunktion zurückgenommen wird.

ExitOpts(MQXEPO)
API-Exits können Optionen angeben, die steuern, wie API-Exits registriert werden. Wenn ein Nullzeiger für dieses Feld angegeben wird, werden die Standardwerte der MQXEPO-Struktur vorausgesetzt.

CompCode (MQLONG) - Ausgabe
Der Beendigungscode, gültige Werte dafür sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ausgabe
Der Ursachencode, der den Beendigungscode qualifiziert.

Wenn der Beendigungscode MQCC_OK ist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn der Beendigungscode MQCC_FAILED ist:

MQRC_HCONFIG_ERROR
(2280, X'8E8') Die bereitgestellte Konfigurationskennung ist nicht gültig. Verwenden Sie die Konfigurationskennung von MQAXP.

MQRC_EXIT_REASON_ERROR
(2377, X'949') Die bereitgestellte Exitfunktion-Aufrufursache ist entweder nicht gültig oder nicht für die bereitgestellte Exitfunktions-ID gültig.

Verwenden Sie entweder eine der gültigen Exitfunktion-Aufrufursachen (MQXR_*-Wert) oder eine gültige Funktions-ID und Exitursachenkombination. (Siehe Tabelle 594 auf Seite 1130.)

MQRC_FUNCTION_ERROR
(2281, X'8E9') Die bereitgestellte Funktions-ID gilt nicht für die API-Exitursache. In der folgenden Tabelle werden die gültigen Kombinationen aus Funktions-IDs und ExitReasons gezeigt.

Tabelle 594. Gültige Kombinationen von Funktions-IDs und ExitReasons

Funktion	ExitReason
MQXF_INIT MQXF_TERM	MQXR_CONNECTION

Tabelle 594. Gültige Kombinationen von Funktions-IDs und ExitReasons (Forts.)	
Funktion	**ExitReason**
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CLOSE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_BEGIN MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_BEFORE MQXR_AFTER
MQXF_DATA_CONV_ON_GET	MQXR_BEFORE

MQRC_RESOURCE_PROBLEM
(2102, X'836') Ein Versuch, eine Exitfunktion zu registrieren oder die Registrierung zurückzunehmen, ist aufgrund eines Ressourcenproblems fehlgeschlagen.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Ein Versuch, eine Exitfunktion zu registrieren oder die Registrierung zurückzunehmen, ist unerwartet fehlgeschlagen.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Ungültiger ExitProperties-Name.

MQRC_XEPO_ERROR
(2507, X'09CB') Exitoptionenstruktur nicht gültig.

MQXEP-Aufruf in Programmiersprache C

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Deklaration für Parameterliste:

```
    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 */
```

MQXEP Funktionsprototyp in C

```
void MQXEP (
MQHCONFIG    Hconfig,        /* Configuration handle */
MQLONG       ExitReason,     /* Exit reason */
```

```
MQLONG       Function,       /* Function identifier */
PMQFUNC      EntryPoint,     /* Function entry point */
PMQXEPO      pExitOpts;      /* Options that control the action of MQXEP */
PMQLONG      pCompCode,      /* Address of completion code */
PMQLONG      pReason);       /* Address of reason code qualifying completion
                                code */
```

Exitfunktionen

In diesem Abschnitt finden Sie allgemeine Informationen zur Verwendung der Funktionsaufrufe sowie eine Erläuterung, wie die einzelnen Exitfunktionen aufgerufen werden.

In diesem Abschnitt finden Sie die allgemeinen Regeln für API-Exitroutinen und erfahren, wie die Exitausführungsumgebung eingerichtet und bereinigt wird.

Allgemeine Regeln für API-Exitroutinen

Beim Aufrufen von API-Exitroutinen gelten die folgenden allgemeinen Regeln.

- In allen Fällen werden die API-Exitfunktionen ausgeführt, bevor API-Aufrufparameter überprüft werden und Sicherheitsprüfungen erfolgen (im Fall von MQCONN, MQCONNX oder MQOPEN).
- Die Werte von Feldern, die bei einer Exitroutine eingegeben oder von ihr ausgegeben werden, sind:
  - Bei Eingabe in eine *Before*-WebSphere MQ API-Exitfunktion kann der Wert eines Feldes durch das Anwendungsprogramm oder durch einen vorherigen Exitfunktionsaufruf eingestellt werden.
  - Bei Ausgabe von einer *Before*-WebSphere MQ API-Exitfunktion kann der Wert eines Feldes unverändert bleiben oder auf einen anderen Wert der Exitfunktion eingestellt werden.
  - Bei Eingabe in eine *After*-WebSphere MQ API-Exitfunktion kann der Wert eines Feldes durch den Warteschlangenmanager nach Verarbeitung des WebSphere MQ API-Aufrufs oder auf einen Wert vor einem vorherigen Exitfunktionsaufruf in der Kette von Exitfunktionen eingestellt werden.
  - Bei Ausgabe von einer *After*-WebSphere MQ API-Aufrufexitfunktion kann der Wert eines Feldes unverändert bleiben oder durch die Exitfunktion auf einen anderen Wert eingestellt werden.
- Exitfunktionen müssen mit dem Warteschlangenmanager mithilfe der Felder ExitResponse und ExitResponse2 kommunizieren.
- Die Felder CompCode und Reason code kommunizieren wiederum mit der Anwendung. Der Warteschlangenmanager und die Exitfunktionen können die Felder CompCode und Reason code einstellen.
- Der MQXEP-Aufruf gibt neue Ursachencodes an die Exitfunktionen zurück, die MQXEP aufrufen. Exitfunktionen können jedoch diese neuen Ursachencodes in einen vorhandenen Ursachencode umsetzen, den vorhandene und neue Anwendungen verstehen können.
- Jeder Exitfunktionsprototyp hat ähnliche Parameter für die API-Funktion mit einer zusätzlichen Zwischenstufe für CompCode und Reason.
- API-Exits können MQI-Aufrufe (außer MQDISC) ausgeben, aber diese MQI-Aufrufe rufen selbst keine API-Exits auf.

Beachten Sie, dass Sie die Reihenfolge der API-Exitaufrufe nicht vorhersagen können, unabhängig davon, ob die Anwendung auf einem Server oder einem Client ausgeführt wird. Auf einen API-Exit-BEF0RE-Aufruf kann möglicherweise nicht sofort ein AFTER-Aufruf folgen.

Auf den BEF0RE-Aufruf kann ein weiterer BEF0RE-Aufruf folgen. Beispiel:

BEFORE MQCTL
BEFORE Callback
BEFORE MQPUT
AFTER MQPUT
AFTER Callback
AFTER MQCTL

oder

BEFORE XAOPEN
BEFORE MQCONNX
AFTER MQCONNX
AFTER XAOPEN

Auf dem Client gibt es einen Exit, der das Verhalten des MQCONN-oder MQCONNX-Aufrufs ändern kann, der als `PreConnect` -Exit bezeichnet wird. Der Exit `PreConnect` kann alle Parameter im MQCONN- oder MQCONNX-Aufruf einschließlich des Warteschlangenmanagernamens ändern. Der Client ruft diesen Exit erst auf und ruft dann den MQCONN- oder MQCONNX-Aufruf ab. Beachten Sie, dass nur der ursprüngliche MQCONN- oder MQCONNX-Aufruf den API-Exit aufruft; alle nachfolgenden Wiederherstellungsaufrufe sind wirkungslos.

Die Ausführungsumgebung

Generell werden alle Fehler von Exitfunktionen mithilfe der Felder ExitResponse und ExitResponse2 in MQAXP zurück an die Exitverwaltung übertragen.

Diese Fehler werden im Gegenzug in MQCC_*- und MQRC_*-Werte umgewandelt und wieder an die Anwendung in den Feldern CompCode und Reason übertragen. Allerdings werden Fehler in der Exitverwaltungslogik wieder zur Anwendung als MQCC_*- und MQRC_*-Werte in den Feldern CompCode und Reason übertragen.

Wenn eine MQ_TERM_EXIT-Funktion einen Fehler zurückgibt:

- Der MQDISC-Aufruf hat bereits stattgefunden
- Es gibt keine weitere Chance, die *After*-MQ_TERM_EXIT-Exitfunktion auszuführen (und somit eine Bereinigung der Exitfunktionausführungsumgebung durchzuführen)
- Die Bereinigung der Exitausführungsumgebung wird *nicht* durchgeführt

Der Exit kann nicht entladen werden, da er möglicherweise noch im Einsatz ist. Auch andere registrierte Exits weiter unten in der Exitkette, für die der *Before*-Exit erfolgreich war, werden in umgekehrter Reihenfolge ausgeführt.

Einrichten der Exitausführungsumgebung

Während der Verarbeitung eines expliziten MQCONN- oder MQCONNX-Aufrufs richtet die Exithandhabungslogik die Exitausführungsumgebung ein, bevor die Exitinitialisierungsfunktion aufgerufen wird (MQ_INIT_EXIT). Zur Einrichtung der Exitausführungsumgebung gehören das Laden des Exits, das Anfordern von Speicherplatz und die Initialisierung von Exitparameterstrukturen. Die Exitkonfigurationskennung wird ebenfalls zugeordnet.

Wenn während dieser Phase Fehler auftreten, schlägt der MQCONN- oder MQCONNX-Aufruf mit CompCode MQCC_FAILED und einem der folgenden Ursachencodes fehl:

MQRC_API_EXIT_LOAD_ERROR
Ein Versuch, ein API-Exitmodul zu laden, ist fehlgeschlagen.

MQRC_API_EXIT_NOT_FOUND
Im API-Exitmodul konnte eine API-Exitfunktion nicht gefunden werden.

MQRC_STORAGE_NOT_AVAILABLE
Ein Versuch, die Ausführungsumgebung für eine API-Exitfunktion zu initialisieren, schlug wegen Speicherknappheit fehl.

MQRC_API_EXIT_INIT_ERROR
Während der Initialisierung der Ausführungsumgebung für eine API-Exitfunktion ist ein Fehler aufgetreten.

Bereinigung der Exitausführungsumgebung

Während der Verarbeitung eines expliziten MQDISC-Aufrufs oder einer impliziten Unterbrechungsanforderung durch eine beendete Anwendung muss die Exithandhabungslogik möglicherweise die Ausfüh-

rungsumgebung bereinigen, nach Aufrufen der Exitbeendigungsfunktion (MQ_TERM_EXIT), falls eine Registrierung vorliegt.

Die Bereinigung der Ausführungsumgebung umfasst die Freigabe von Speicherbereich für Exitparameterstrukturen, wodurch möglicherweise vorher in den Speicher geladene Module gelöscht werden.

Falls Fehler während dieser Phase auftreten, schlägt ein expliziter MQDISC-Aufruf mit CompCode MQCC_FAILED und folgendem Ursachencode fehl (Fehler werden nicht bei impliziten Unterbrechungsanforderungen hervorgehoben):

MQRC_API_EXIT_TERM_ERROR
Beim Schließen der Ausführungsumgebung für eine API-Exitfunktion ist ein Fehler aufgetreten. Der Exit sollte vor oder nach den MQ_TERM* API-Exitfunktionsaufrufen *keine* Fehler von der MQDISC zurückgeben.

API-Exits bei Clients

Ein Client verwendet den PreConnect-Exit, um das Verhalten der MQCONN- und MQCONNX-Aufrufe zu ändern, und unterstützt keine API-Exiteigenschaften.

PreConnect-Exit

Bei einem Client kann der PreConnect-Exit verwendet werden, um die Kanaldefinition von einem zentralen Repository, z. B. einem LDAP-Server, aus zu suchen.

Der PreConnect-Exit kann außerdem alle Parameter oder alle Parameter bei einem MQCONN- oder MQCONNX-Aufruf selbst ändern, z. B. den Warteschlangenmanagername.

Im Fall von Clientanwendungen muss der PreConnect-Exit vor dem API-Exit aufgerufen werden, weil der MQCONN- oder MQCONNX-API-Exit erst aufgerufen wird, wenn der Name des Warteschlangenmanagers bekannt ist. Dieser Name kann vom PreConnect-Exit geändert werden.

Beachten Sie, dass nur der ursprüngliche MQCONN- oder MQCONNX-Aufruf den Exit aufruft.

API-Exiteigenschaften

Auf einem Server können API-Exits eine MQXEPO-Struktur zur Initialisierungszeit registrieren. Die MQXEPO-Struktur enthält das Feld ExitProperties, das Details zur Gruppe der Eigenschaften aufführt, an denen der Exit interessiert ist. Dadurch wird eine separate Nachrichteneigenschaftskennung erstellt, die der Exit getrennt von allen Eigenschaftskennungen der Anwendungsnachrichten bearbeiten kann.

Auf einem Client werden API-Exiteigenschaften nicht unterstützt. Wenn versucht wird, einen Eigenschaftsgruppennamen auf einem Client zu registrieren, schlägt die Funktion mit dem Ursachencode MQRC_EXIT_PROPS_NOT_SUPPORTED fehl.

Rücksetzung - MQ_BACK_EXIT

MQ_BACK_EXIT stellt eine Rücksetzungexitfunktion zum Ausführen einer *Before*- und *After*-Rücksetzungsverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_BACK mit Exitfunktionen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-Rücksetzungsaufruf-Exitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Anfang - MQ_BEGIN_EXIT

MQ_BEGIN_EXIT stellt eine Anfangsexitfunktion zum Ausführen einer *Before*- und *After*-MQBEGIN-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_BEGIN mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before* und *After*-MQBEGIN-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pBeginOptions (PMQBO)- Ein-/Ausgabe
Verweis auf Anfangsoptionen.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PPMQBO    ppBeginOptions,   /* Address of ptr to begin options */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Callback - MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT stellt eine Exitfunktion zur Verfügung, mit der eine Verarbeitung zur *Vorbereitung* und *Nachbereitung* eines Callbacks ausgeführt wird. Verwenden Sie die Funktions-ID MQXF_CALLBACK mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von Exitfunktionen zur *Vorbereitung* und *Nachbereitung* eines Callback-Aufrufs.

Die Schnittstelle für diese Funktion ist:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
    &pBuffer, &pMQCBContext)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Ein-/Ausgabe
Verbindungskennung

pMsgDesc
Nachrichtendeskriptor

pGetMsgOpts
Optionen zur Steuerung der Aktion von MQGET

pBuffer
Bereich, der die Nachrichtendaten enthält

pMQCBContext
Kontextdaten für den Callback

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
 &pContext);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP    pExitParms;    /* Exit parameter structure */
PMQAXC    pExitContext;  /* Exit context structure */
PMQHCONN  pHconn;        /* Connection handle */
PPMQMD    ppMsgDesc;     /* Message descriptor */
PPMQGMO   ppGetMsgOpts;  /* Options that define the operation of the consumer */
PPMQVOID  ppBuffer;      /* Area to conatin the message data */
PPMQCBC   ppContext;)     /* Context data for the callback */
```

Hinweise zur Verwendung

1. Der Rückrufexit wird aufgerufen, bevor der Konsument aufgerufen wird und nachdem dessen Konsumentenfunktion abgeschlossen wurde. Die MQMD- und die MQGMO-Struktur sind zwar veränderbar, doch bewirkt die Änderung der Werte im Before-Exit keinen Abruf einer Nachricht aus der Warteschlange, da die Nachricht bereits aus der Warteschlange entfernt wurde, die an die Konsumentenfunktion übermittelt werden soll.

Callback-Funktionen steuern - MQ_CB_EXIT

MQ_CB_EXIT stellt eine Exitfunktion zur Verfügung, die *vor* und *nach* dem MQCB-Aufruf ausgeführt wird. Verwenden Sie die Funktions-ID MQXF_CB mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von Exitfunktionen zur *Vorbereitung* und *Nachbereitung* eines MQCB-Aufrufs.

Die Schnittstelle für diese Funktion ist:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Ein-/Ausgabe
Verbindungskennung

Operation (MQLONG) - Ein-/Ausgabe
Operationswert

pCallbackDesc (PMQCBD) - Ein-/Ausgabe
Callback-Deskriptor

Hobj (MQHOBJ) - Ein-/Ausgabe
Objektkennung

pMsgDesc (PMQMD) - Ein-/Ausgabe
Nachrichtendeskriptor

pGetMsgOpts (PMQGMO) - Ein-/Ausgabe
Optionen zur Steuerung der Aktion von MQCB

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode (CompCode) näher bestimmt wird

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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.
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
        MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
                    &pGetMsgOpts, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Schließen - MQ_CLOSE_EXIT

MQ_CLOSE_EXIT stellt eine Schließexitfunktion zum Ausführen einer *Before*- und *After*-MQCLOSE-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_CLOSE mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQCLOSE-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pHobj (PMQHOBJ) - Eingabe
Verweis auf Objektkennung.

Options (MQLONG)- Ein-/Ausgabe
Schließoptionen.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Lautet der Beendigungscode MQCC_FAILED, kann die Exitfunktion im Feld für den Ursachencode einen beliebigen gültigen Wert MQRC_* festlegen.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
void MQENTRY MQ_CLOSE_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQHOBJ        ppHobj,         /* Address of ptr to object handle */
PMQLONG         pOptions,       /* Address of close options */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Commit - MQ_CMIT_EXIT

MQ_CMIT_EXIT stellt eine Commitexitfunktion zum Ausführen einer *Before*- und *After*-Commitverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_CMIT mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von *Before*- und *After*-Commitaufrufexitfunktionen.

Wenn eine Commitoperation fehlschlägt und die Transaktion zurückgesetzt wird, schlägt der MQCMIT-Aufruf mit MQCC_WARNING und MQRC_BACKED_OUT fehl. Diese Rückgabe- und Ursachencodes werden an alle *After*-MQCMIT-Exitfunktionen übergeben, um den Exitfunktionen zu melden, dass die Arbeitseinheit zurückgesetzt wurde.

Die Schnittstelle für diese Funktion ist:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

1. Die hier beschriebene Schnittstelle für die Funktion MQ_GET_EXIT wird sowohl für die Exitfunktion MQXF_GET als auch für die Exitfunktion „MQXF_DATA_CONV_ON_GET" auf Seite 1147 verwendet.

   Für diese beiden Exitfunktionen sind separate Eingangspunkte definiert. Zum Abfangen *beider* Exitfunktionen muss daher der MQXEP-Aufruf zweimal verwendet werden. Verwenden Sie für diesen Aufruf die Funktions-ID MQXF_GET.

   Da die Schnittstelle von MQ_GET_EXIT für MQXF_GET und MQXF_DATA_CONV_ON_GET identisch ist, kann für die beiden Aufrufe eine einzelne Exitfunktion verwendet werden. Das Feld *Function* in der MQAXP-Struktur gibt an, welche Exitfunktion aufgerufen wurde. Alternativ kann der MQXEP-Aufruf verwendet werden, um für die beiden Fälle andere Exitfunktionen zu registrieren.

Verbindung herstellen und erweitern - MQ_CONNX_EXIT

MQ_CONNX_EXIT stellt Folgendes bereit:

- Verbindungsexitfunktion für die Ausführung einer *Before*- und *After*-MQCONN-Verarbeitung
- Verbindungserweiterungsexitfunktionen für die Ausführung einer *Before*- und *After*-MQCONNX-Verarbeitung

Dieselbe, hier beschriebene Schnittstelle wird von MQCONN- und MQCONNX-Aufrufexitfunktionen aufgerufen.

Wenn der Nachrichtenkanalagent einer eingehenden Clientverbindung antwortet, kann der Nachrichtenkanalagent eine Verbindung herstellen und mehrere WebSphere MQ API-Aufrufe ausführen, bevor der Clientstatus vollständig bekannt ist. Die API-Aufrufe rufen die API-Exitfunktionen mit dem auf dem Nachrichtenkanalagentenprogramm selbst basierenden MQAXC auf (z. B. in den Feldern UserId und ConnectionName von MQAXC).

Wenn der Nachrichtenkanalagent den nachfolgenden Client-API-Aufrufen antwortet, basiert die MQAXC-Struktur auf dem eingehenden Client, der die Felder UserId und ConnectionName entsprechend einstellt.

Der von der Anwendung auf einem MQCONN- oder MQCONNX-Aufruf eingestellte Warteschlangenmanagername wird an den zugrunde liegenden Aufruf übergeben. Versuche, den Namen des Warteschlangenmanagers durch einen *Before*-MQ_CONNX_EXIT zu ändern, bleiben wirkungslos.

Verwenden Sie bei den Exitursachen MQXR_BEFORE und MQXR_AFTER die Funktions-IDs MQXF_CONN und MQXF_CONNX, um *Before*- und *After*-MQCONN- und MQCONNX-Aufrufexitfunktionen zu registrieren.

Ein MQ_CONNX_EXIT-Exit, der für Ursache MQXR_BEFORE aufgerufen wurde, *darf keine* WebSphere MQ API-Aufrufe ausgeben, da die korrekte Umgebung zu diesem Zeitpunkt nicht eingerichtet war.

Ein MQ_CONNX_EXIT kann MQDISC nicht über einen API-Exit-Aufruf für die Verbindung aufrufen, auf die sich der Aufruf bezieht. Diese Einschränkung gilt sowohl für Client- als auch für Server-API-Exits.

Die Schnittstelle zu MQCONN und MQCONNX ist identisch:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
               &pHconn, &CompCode, &Reason);
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

pQMgrName (PMQCHAR) - Eingabe
Verweis auf den Warteschlangenmanagernamen, der beim MQCONNX-Aufruf bereitgestellt wurde. Der Exit darf diesen Namen weder beim MQCONN- noch beim MQCONNX-Aufruf ändern.

pConnectOpts (PMQCNO) - Ein-/Ausgabe
Verweis auf die Optionen, die die Maßnahme des MQCONNX-Aufrufs kontrollieren.

Details siehe „MQCNO - Verbindungsoptionen” auf Seite 300.

Für die Exitfunktion MQXF_CONN verweist pConnectOpts auf die Struktur der Standardverbindungsoptionen (MQCNO_DEFAULT).

pHconn (PMQHCONN) - Eingabe
Verweis auf die Verbindungskennung.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung).

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOps,
               &pHconn, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

1. Die hier beschriebene Schnittstelle für die Funktion MQ_CONNX_EXIT wird sowohl für den Aufruf MQCONN als auch MQCONNX verwendet. Allerdings sind für diese beiden Aufrufe separate Eingangspunkte definiert. Zum Abfangen *beider* Aufrufe muss der MQXEP-Aufruf mindestens zweimal verwendet werden – einmal mit der Funktions-ID MQXF_CONN, das zweite Mal mit MQXF_CONNX.

   Da die Schnittstelle von MQ_CONNX_EXIT für MQCONN und MQCONNX identisch ist, kann für die beiden Aufrufe eine einzelne Exitfunktion verwendet werden. Das Feld *Function* in der MQAXP-Struktur

gibt an, welcher Aufruf gerade in Bearbeitung ist. Alternativ dazu können mit dem MQXEP-Aufruf verschiedene Exitfunktionen für die beiden Aufrufe registriert werden.

2. Wenn ein Nachrichtenkanalagent (MCA) auf eine eingehende Clientverbindung antwortet, kann er eine Reihe von MQ-Aufrufen absetzen, bevor der Clientstatus vollständig bekannt ist. Diese MQ-Aufrufe bewirken, dass die API-Exitfunktionen mit der MQAXC-Struktur aufgerufen werden, die keine Daten zum Client, sondern zum MCA enthalten (beispielsweise Benutzer-ID und Verbindungsname). Sobald jedoch der Clientstatus vollständig bekannt ist, bewirken nachfolgende MQ-Aufrufe, dass die API-Exitfunktionen mit den entsprechenden Clientdaten in der MQAXC-Struktur aufgerufen werden.
3. Es werden alle MQXR_BEFORE-Exitfunktionen aufgerufen, bevor der Warteschlangenmanager eine Gültigkeitsprüfung der Parameter ausführt. Daher sind die Parameter möglicherweise ungültig (einschließlich ungültiger Verweise auf die Adressen der Parameter).

   Die Funktion MQ_CONNX_EXIT wird aufgerufen, bevor der Warteschlangenmanager irgendwelche Berechtigungsprüfungen vornimmt.
4. Die Exitfunktion darf den im MQCONN- oder MQCONNX-Aufruf angegebenen Namen des Warteschlangenmanagers nicht ändern. Wenn die Exitfunktion den Namen ändert, sind die Ergebnisse nicht definiert.
5. Eine MQXR_BEFORE-Exitfunktion für MQ_CONNX_EXIT kann keine anderen MQ-Aufrufe als MQXEP absetzen.

Callback-Steuerung - MQ_CTL_EXIT

MQ_CTL_EXIT stellt eine Exitfunktion für Subskriptionsanforderungen zur Verfügung, mit der eine Verarbeitung zur *Vorbereitung* und *Nachbereitung* einer Callback-Steuerung ausgeführt wird. Verwenden Sie die Funktions-ID MQXF_CTL mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von Exitfunktionen zur *Vorbereitung* und *Nachbereitung* eines Aufrufs der Callback-Steuerung.

Die Schnittstelle für diese Funktion ist:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

Hconn (MQHCONN) - Ein-/Ausgabe
Verbindungskennung.

Operation (MQLONG) Ein-/Ausgabe
Die Operation, die für den Callback verarbeitet wird, der für die angegebene Objektkennung definiert wurde

ControlOpts (MQCTLO) Ein-/Ausgabe
Optionen zur Steuerung der Aktion von MQCTL

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Unterbrechen - MQ_DISC_EXIT

MQ_DISC_EXIT stellt eine Unterbrechungsexitfunktion zum Ausführen einer *Before*- und *After*-MQDISC-Exitverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_DISC mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von *Before*- und *After*-MQDISC-Aufrufexitfunktionen.

Die Schnittstelle für diese Funktion ist

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

pHconn (PMQHCONN) - Eingabe
Verweis auf die Verbindungskennung.

Beim Before-MQDISC-Aufruf ist der Wert dieses Feldes einer der Folgenden:

- Die beim MQCONN- oder MQCONNX-Aufruf zurückgegebene Verbindungskennung
- Null, für Umgebungen, wo ein umgebungsspezifischer Adapter eine Verbindung mit dem Warteschlangenmanager hergestellt hat
- Ein Wert, der von einem vorherigen Exitfunktionsaufruf eingestellt wurde

Beim After-MQDISC-Aufruf ist der Wert dieses Felds Null oder ein Wert, der von einem vorherigen Exitfunktionsaufruf eingestellt wurde.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Ausführung

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
PMQHCONN        pHconn;         /* Ptr to Connection handle */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Abrufen - MQ_GET_EXIT

MQ_GET_EXIT stellt eine Abrufexitfunktion für die Ausführung der *Vor* und *Nach* MQGET-Aufrufverarbeitung bereit.

Es gibt zwei Funktions-IDs:

1. Verwenden Sie MQXF_GET mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQGET-Aufrufexitfunktionen zu registrieren.
2. Im Abschnitt „MQXF_DATA_CONV_ON_GET" auf Seite 1147 finden Sie Informationen über die Verwendung der MQXF_DATA_CONV_ON_GET-Funktions-ID.

Die Schnittstelle für diese Funktion ist:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

Hobj (MQHOBJ) - Ein-/Ausgabe
Objektkennung.

pMsgDesc (PMQMD) - Ein-/Ausgabe
Verweis auf Nachrichtendeskriptor.

pGetMsgOpts (PMQGMO) - Ein-/Ausgabe
Verweis auf die Nachrichtenabrufoptionen.

BufferLength (MQLONG) - Ein-/Ausgabe
Nachrichtenpufferlänge.

pBuffer (PMQBYTE) - Ein-/Ausgabe
Verweis auf Nachrichtenpuffer.

pDataLength (PMQLONG) - Ein-/Ausgabe
Verweis auf Datenlängenfeld.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

1. Die hier beschriebene Schnittstelle für die Funktion MQ_GET_EXIT wird sowohl für die Exitfunktion MQXF_GET als auch für die Exitfunktion „MQXF_DATA_CONV_ON_GET” auf Seite 1147 verwendet.

   Für diese beiden Exitfunktionen sind separate Eingangspunkte definiert. Zum Abfangen *beider* Exitfunktionen muss daher der MQXEP-Aufruf zweimal verwendet werden. Verwenden Sie für diesen Aufruf die Funktions-ID MQXF_GET.

   Da die Schnittstelle von MQ_GET_EXIT für MQXF_GET und MQXF_DATA_CONV_ON_GET identisch ist, kann für die beiden Aufrufe eine einzelne Exitfunktion verwendet werden. Das Feld *Function* in der MQAXP-Struktur gibt an, welche Exitfunktion aufgerufen wurde. Alternativ kann der MQXEP-Aufruf verwendet werden, um für die beiden Fälle andere Exitfunktionen zu registrieren.

MQXF_DATA_CONV_ON_GET

Unter MQ_GET_EXIT finden Sie Informationen zur Schnittstelle für diesen Aufruf sowie eine Beispieldeklaration in der Programmiersprache C.

Hinweise zur Verwendung

Falls er registriert ist, wird dieser Eingangspunkt aufgerufen, wenn bei der Anwendung Nachrichten eingehen, aber bevor eine Datenkonvertierung aufgetreten ist. Dies ist hilfreich, wenn der API-Exit einen Verarbeitungsvorgang ausführt, z. B. Entschlüsselung oder Dekomprimierung, bevor die Nachricht an die Datenkonvertierung übergeben wird. Der Exit kann, falls erforderlich, veranlassen, dass die Datenkonvertierung durch Rückgabe von MQXCC_SUPPRESS_FUNCTION umgangen wird; weitere Informationen erhalten Sie im Abschnitt zur MQAXP-Struktur.

Die Registrierung für diesen Eingangspunkt bei einem Client bewirkt, dass die Datenkonvertierung lokal auf dem Clientsystem ausgeführt wird. Für einen ordnungsgemäßen Betrieb kann es daher erforderlich sein, die Anwendungskonvertierungsexits auf dem Client zu installieren. Beachten Sie, dass MQXF_DATA_CONV_ON_GET außerdem für asynchrone Nutzung verwendet werden kann.

Wenn Sie den MQ_GET_EXIT-Aufruf verwenden, verwenden Sie MQXF_DATA_CONV_ON_GET mit der Exitursache MQXR_BEFORE, um eine Exitfunktion für die *Before*-MQGET-Datenkonvertierung zu registrieren.

Es gibt keine MQXR_AFTER-Exitfunktion für MQXF_DATA_CONV_ON_GET; die MQXR_AFTER-Exitfunktion für MQXF_GET stellt die für die Exitverarbeitung nach der Datenkonvertierung erforderlichen Funktionen bereit.

Separate Eingangspunkte werden für den MQ_GET_EXIT-Aufruf definiert, sodass der MQXEP-Aufruf zweimal verwendet werden muss, damit *beide* Exitfunktionen abgefangen werden; verwenden Sie für diesen Aufruf die Funktions-ID MQXF_DATA_CONV_ON_GET.

Da die Schnittstelle von MQ_GET_EXIT für MQXF_GET und MQXF_DATA_CONV_ON_GET identisch ist, kann für die beiden Aufrufe eine einzelne Exitfunktion verwendet werden. Das Feld *Function* in der MQAXP-Struktur gibt an, welche Exitfunktion aufgerufen wurde. Alternativ kann der MQXEP-Aufruf verwendet werden, um für die beiden Fälle andere Exitfunktionen zu registrieren.

Initialisierung - MQ_INIT_EXIT

MQ_INIT_EXIT stellt die Initialisierung der Verbindungsebene bereit, die durch Setzen von ExitReason in MQAXP auf MQXR_CONNECTION angegeben wird.

Beachten Sie während der Initialisierung Folgendes:

- Die MQ_INIT_EXIT-Funktion ruft MQXEP auf, um die WebSphere MQ API-Verben und die ENTRY- und EXIT-Punkte zu registrieren, an denen sie interessiert ist.
- Exits müssen nicht alle WebSphere MQ API-Verben abfangen. Exitfunktionen werden nur aufgerufen, wenn ein Anspruch registriert wurde.

- Speicher, der vom Exit verwendet werden soll, kann während der der Initialisierung angefordert werden.
- Wenn ein Aufruf an diese Funktion fehlschlägt, schlägt der MQCONN- oder MQCONNX-Aufruf, der sie aufgerufen hat, mit einem CompCode und Reason, der vom Wert des ExitResponse-Felds in MQAXP abhängig ist, ebenfalls fehl.
- Ein MQ_INIT_EXIT-Exit darf keine WebSphere MQ API-Aufrufe ausgeben, weil die richtige Umgebung zu diesem Zeitpunkt nicht eingerichtet wurde.
- Wenn ein MQ_INIT_EXIT mit MQXCC_FAILED fehlschlägt, kehrt der Warteschlangenmanager vom MQCONN- oder MQCONNX-Aufruf zurück, der ihn mit MQCC_FAILED und MQRC_API_EXIT_ERROR aufgerufen hat.
- Wenn der Warteschlangenmanager bei der Initialisierung der Ausführungsumgebung der API-Exitfunktion einen Fehler feststellt, bevor der erste MQ_INIT_EXIT aufgerufen wird, kehrt der Warteschlangenmanager vom MQCONN- oder MQCONNX-Aufruf zurück, der MQ_INIT_EXIT mit MQCC_FAILED und MQRC_API_EXIT_INIT_ERROR aufgerufen hat.

Die Schnittstelle zu MQ_INIT_EXIT ist:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

CompCode (MQLONG) - Ein-/Ausgabe
Verweis auf Beendigungscode, gültige Werte dafür sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Verweis auf Ursachencode zur Qualifikation des Beendigungscodes.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Der an die Anwendung zurückgegebene CompCode und Reason hängt vom Wert des ExitResponse-Felds in MQAXP ab.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

1. Die MQ_INIT_EXIT-Funktion kann den MQXEP-Aufruf absetzen, um die Adressen der Exitfunktionen für die abzufangenden MQ-Aufrufe zu registrieren. Es ist nicht notwendig, alle MQ-Aufrufe abzufangen oder sowohl MQXR_BEFORE- als auch MQXR_AFTER-Aufrufe abzufangen. Zum Beispiel könnte eine Exit-Suite auswählen, dass nur der Aufruf MQXR_BEFORE von MQPUT abgefangen wird.
2. Der von Exitfunktionen in der Exit-Suite zu verwendende Speicherplatz kann mit der Funktion MQ_INIT_EXIT angefordert werden. Alternativ können Exitfunktionen bei ihrem Aufruf Speicherplatz anfordern, wenn dies erforderlich ist. Allerdings sollte der gesamte Speicherplatz vor dem Beenden der Exit-Suite freigegeben werden. Die Funktion MQ_TERM_EXIT oder eine früher aufgerufene Exitfunktion kann den Speicherplatz freigeben.
3. Wenn MQ_INIT_EXIT im Feld *ExitResponse* von MQAXP den Wert MQXCC_FAILED zurückgibt oder auf andere Weise fehlschlägt, schlägt auch der MQCONN- oder MQCONNX-Aufruf fehl, der den Aufruf von MQ_INIT_EXIT bewirkt hat, wobei die Parameter *CompCode* und *Reason* auf die entsprechenden Werte eingestellt sind.
4. Eine MQ_INIT_EXIT-Funktion kann keine anderen MQ-Aufrufe als MQXEP absetzen.

Abfragen - MQ_INQ_EXIT

MQ_INQ_EXIT stellt eine Abfrageexitfunktion zum Ausführen einer *Before*- und *After*-MQINQ-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_INQ mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQINQ-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

Hobj (MQHOBJ) - Eingabe
Objektkennung.

SelectorCount (MQLONG) - Eingabe
Selektorenanzahl

pSelectors (PMQLONG) - Ein-/Ausgabe
Verweis auf Feldgruppe von Selektorwerten.

IntAttrCount (MQLONG) - Eingabe
Anzahl Ganzzahlenattribute.

pIntAttrs (PMQLONG) - Ein-/Ausgabe
Verweis auf Attributwert der Ganzzahlenfeldgruppe.

CharAttrLength (MQLONG) - Ein-/Ausgabe
Feldgruppenlänge der Zeichenattribute.

pCharAttrs (PMQCHAR) - Ein-/Ausgabe
Verweis auf Zeichenattributfeldgruppe.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Öffnen - MQ_OPEN_EXIT

MQ_OPEN_EXIT stellt eine Öffnungsexitfunktion zum Ausführen einer *Before*- und *After*-MQOPEN-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_OPEN mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQOPEN-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pObjDesc (PMQOD) - Ein-/Ausgabe
Verweis aus Objektdeskriptor.

Options (MQLONG) - Ein-/Ausgabe
Öffnungsoptionen.

pHobj (PMQHOBJ) - Eingabe
Verweis auf Objektkennung.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Ausführung

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Einreihen - MQ_PUT_EXIT

MQ_PUT_EXIT stellt eine Einreihungsexitfunktion zum Ausführen einer *Before*- und *After*-MQPUT-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_PUT mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After* MQPUT-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

Hobj (MQHOBJ) - Ein-/Ausgabe
Objektkennung.

pMsgDesc (PMQMD) - Ein-/Ausgabe
Verweis auf Nachrichtendeskriptor.

pPutMsgOpts (PMQPMO) - Ein-/Ausgabe
Verweis auf Nachrichteneinreihungsoptionen.

BufferLength (MQLONG) - Ein-/Ausgabe
Nachrichtenpufferlänge.

pBuffer (PMQBYTE) - Ein-/Ausgabe
Verweis auf Nachrichtenpuffer.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

- Vom Warteschlangenmanager erstellte Berichtsnachrichten überspringen die normale Aufrufverarbeitung. Dadurch können solche Nachrichten nicht von der Funktion MQ_PUT_EXIT oder MQPUT1 abgefangen werden. Vom Nachrichtenkanalagenten erstellte Berichtsnachrichten werden dagegen normal verarbeitet und können daher von der Funktion MQ_PUT_EXIT oder MQ_PUT1_EXIT abgefangen werden. Um sicherzustellen, dass alle vom Nachrichtenkanalagenten erstellten Berichtsnachrichten angefangen werden, sollte sowohl MQ_PUT_EXIT als auch MQ_PUT1_EXIT verwendet werden.

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT ermöglicht eine Exitfunktion zum *Einreihen nur einer Nachricht*, um eine *Before*- und *After*-MQPUT1-Aufrufverarbeitung auszuführen. Verwenden Sie die Funktions-ID MQXF_PUT1 mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQPUT1-Aufrufexitfunktionen auszuführen.

Die Schnittstelle für diese Funktion ist:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pObjDesc (PMQOD) - Ein-/Ausgabe
Verweis aus Objektdeskriptor.

pMsgDesc (PMQMD) - Ein-/Ausgabe
Verweis auf Nachrichtendeskriptor.

pPutMsgOpts (PMQPMO) - Ein-/Ausgabe
Verweis auf Nachrichteneinreihungsoptionen.

BufferLength (MQLONG) - Ein-/Ausgabe
Nachrichtenpufferlänge.

pBuffer (PMQBYTE) - Ein-/Ausgabe
Verweis auf Nachrichtenpuffer.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Einstellen - MQ_SET_EXIT

MQ_SET_EXIT stellt eine Einstellungsexitfunktion zum Ausführen einer *Before*- und *After*-MQSET-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_SET mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQSET-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

Hobj (MQHOBJ) - Eingabe
Objektkennung.

SelectorCount (MQLONG) - Eingabe
Selektorenanzahl

pSelectors (PMQLONG) - Ein-/Ausgabe
Verweis auf Feldgruppe von Selektorwerten.

IntAttrCount (MQLONG) - Eingabe
Anzahl Ganzzahlenattribute.

pIntAttrs (PMQLONG) - Ein-/Ausgabe
Verweis auf Attributwert der Ganzzahlenfeldgruppe.

CharAttrLength (MQLONG) - Ein-/Ausgabe
Feldgruppenlänge der Zeichenattribute.

pCharAttrs (PMQCHAR) - Ein-/Ausgabe
Verweis auf Zeichenattributwerte.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
void MQENTRY MQ_SET_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQHOBJ   pHobj,            /* Address of object handle */
PMQLONG   pSelectorCount,   /* Address of selector count */
PPMQLONG  ppSelectors,      /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;    /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,       /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,  /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,      /* Address of ptr to character attributes array */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Status - MQ_STAT_EXIT

MQ_STAT_EXIT stellt eine Statusexitfunktion zum Ausführen einer *Before*- und *After* MQSTAT-Aufrufverarbeitung bereit. Verwenden Sie die Funktions-ID MQXF_STAT mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um *Before*- und *After*-MQSTAT-Aufrufexitfunktionen zu registrieren.

Die Schnittstelle für diese Funktion ist:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

Type (MQLONG) - Eingabe
Abzurufender Typ von Statusinformationen.

pStatus (PMQSTS) - Ausgabe
Verweis auf Statuspuffer.

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Beendigung - MQ_TERM_EXIT

MQ_TERM_EXIT ermöglicht Beendigung auf Verbindungsebene, registriert mit einer Funktions-ID von MQXF_TERM und ExitReason MQXR_CONNECTION. Wenn registriert, wird MQ_TERM_EXIT einmal für jede Anforderung zum Trennen der Verbindung aufgerufen.

Als Teil der Beendigung kann der nicht mehr erforderliche Speicher vom Exit freigegeben werden und es kann die erforderliche Bereinigung erfolgen.

Wenn ein MQ_TERM_EXIT mit MQXCC_FAILED fehlschlägt, kehrt der Warteschlangenmanager vom MQDISC-Aufruf zurück, der ihn mit MQCC_FAILED und MQRC_API_EXIT_ERROR aufgerufen hat.

Wenn der Warteschlangenmanager bei der Beendigung der Ausführungsumgebung der API-Exitfunktion einen Fehler feststellt, nachdem der letzte MQ_TERM_EXIT aufgerufen wird, kehrt der Warteschlangenmanager vom MQDISC- oder MQCONNX-Aufruf zurück, der MQ_TERM_EXIT mit MQCC_FAILED und MQRC_API_EXIT_TERM_ERROR aufgerufen hat.

Die Schnittstelle für diese Funktion ist:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Lautet der Beendigungscode MQCC_FAILED, kann die Exitfunktion im Feld für den Ursachencode einen beliebigen gültigen Wert MQRC_* festlegen.

Der an die Anwendung zurückgegebene CompCode und Reason hängt vom Wert des ExitResponse-Felds in MQAXP ab.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Hinweise zur Verwendung

1. Die Funktion MQ_TERM_EXIT ist optional. Es ist nicht notwendig, dass eine Exit-Suite einen Beendigungsexit registriert, wenn keine Beendigungsverarbeitung erfolgen muss.

   Wenn Funktionen, die zu der Exit-Suite gehören, während eines Verbindungsaufbaus Ressourcen anfordern, ist eine MQ_TERM_EXIT-Funktion eine bequeme Stelle für die Freigabe solcher Ressourcen, beispielsweise durch die Freigabe von dynamisch zugeordnetem Speicherplatz.
2. Wird bei Ausgabe des MQDISC-Aufrufs eine MQ_TERM_EXIT-Funktion registriert, wird diese nach Ausführung aller MQDISC-Exitfunktionen aufgerufen.
3. Wenn MQ_TERM_EXIT im Feld *ExitResponse* von MQAXP den Wert MQXCC_FAILED zurückgibt oder auf andere Weise fehlschlägt, schlägt auch der MQDISC-Aufruf fehl, der den Aufruf von MQ_TERM_EXIT bewirkt hat, wobei die Parameter *CompCode* und *Reason* auf die entsprechenden Werte eingestellt sind.

Subskription registrieren - MQ_SUB_EXIT

MQ_SUB_EXIT stellt eine Exitfunktion zur Verfügung, mit der eine Verarbeitung zur *Vorbereitung* und *Nachbereitung* einer erneuten Registrierung einer Subskription ausgeführt wird. Verwenden Sie die Funktions-ID MQXF_SUB mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von Exitfunktionen zur *Vorbereitung* und *Nachbereitung* eines Aufrufs zur erneuten Registrierung einer Subskription.

Die Schnittstelle für diese Funktion ist:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Ein-/Ausgabe
Verbindungskennung.

pSubDesc - Ein-/Ausgabe
Feldgruppe aus Attributselektoren.

pHobj - Ein-/Ausgabe
Objektkennung

pHsub (MQHOBJ) Ein-/Ausgabe
Subskriptionskennung

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
 &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
        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 */
```

Subskriptionsanforderung - MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT stellt eine Exitfunktion für Subskriptionsanforderungen zur Verfügung, mit der eine Verarbeitung zur *Vorbereitung* und *Nachbereitung* von Subskriptionsanforderungen ausgeführt wird. Verwenden Sie die Funktions-ID MQXF_SUBRQ mit den Exitursachen MQXR_BEFORE und MQXR_AFTER für die Registrierung von Exitfunktionen zur *Vorbereitung* und *Nachbereitung* eines Subskriptionsanforderungsaufrufs.

Die Schnittstelle für diese Funktion ist:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
    &CompCode, &Reason)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Ein-/Ausgabe
Verbindungskennung.

pHsub (MQHOBJ) Ein-/Ausgabe
Subskriptionskennung

Action (MQLONG) Ein-/Ausgabe
Action

pSubRqOpts (MQSRO) Ein-/Ausgabe

CompCode (MQLONG) - Ein-/Ausgabe
Beendigungscode, für den folgende Werte gültig sind:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason (MQLONG) - Ein-/Ausgabe
Ursachencode, mit dem der Beendigungscode näher bestimmt wird.

Lautet der Beendigungscode MQCC_OK, ist nur der folgende Wert gültig:

MQRC_NONE
(0, x'000') Keine zu meldende Ursache vorhanden.

Wenn der Beendigungscode MQCC_FAILED oder MQCC_WARNING lautet, kann das Feld für den Ursachencode von der Exitfunktion auf einen beliebigen MQRC_*-Wert gesetzt werden.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
        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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
 &CompCode, &Reason);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_close-Exitfunktion bereit, die vor und nach einer xa_close-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XACLOSE zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_close-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXa_info (PMQCHAR) - Ein-/Ausgabe
Instanzenspezifische Ressourcenmanagerdaten.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_commit-Exitfunktion bereit, die vor und nach der xa_commit-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XACOMMIT zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_commit-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_complete-Exitfunktion bereit, die vor und nach einer xa_complete-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XACOMPLETE zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_complete-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags, &XARetC
ode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pHandle (PMQLONG) - Ein-/Ausgabe
Verweis auf asynchrone Operation.

pRetVal (PMQLONG) - Ein-/Ausgabe
Rückgabewert der asynchronen Operation.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags, &XARetC
ode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_end-Exitfunktion bereit, die vor und nach der xa_end-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XAEND zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_end-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
typedef void MQENTRY XA_END_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_forget - XA_FORGET_EXIT

XA_FORGET_EXIT stellt eine xa_forget-Exitfunktion bereit, die vor und nach einer xa_forget-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XAFORGET zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_forget-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_open-Exitfunktion bereit, die vor und nach der xa_open-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XAOPEN zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_open-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXa_info (PMQCHAR) - Ein-/Ausgabe
Instanzenspezifische Ressourcenmanagerdaten.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_prepare-Exitfunktion bereit, die vor und nach einer xa_prepare-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XAPREPARE zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_prepare-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_recover-Exitfunktion bereit, die vor und nach der xa_recover-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XARECOVER zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_recover-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Count (MQLONG) - Ein-/Ausgabe
Maximale Anzahl der XIDs im XID-Bereich.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_rollback-Exitfunktion bereit, die vor und nach einer xa_rollback-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XAROLLBACK zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_rollback-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine xa_start-Exitfunktion bereit, die vor und nach der xa_start-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_XASTART zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem xa_start-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine ax_reg-Exitfunktion bereit, die vor und nach der ax_reg-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_AXREG zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem ax_reg-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Hconn (MQHCONN) - Eingabe
Verbindungskennung.

pXID (MQPTR) - Ein-/Ausgabe
Transaktionsverzweigungs-ID.

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 stellt eine ax_unreg-Exitfunktion bereit, die vor und nach der ax_unreg-Verarbeitung ausgeführt werden soll. Verwenden Sie die Funktions-ID MQXF_AXUNREG zusammen mit den Exitursachen MQXR_BEFORE und MQXR_AFTER, um die Exitfunktionen zu registrieren, die vor und nach einem ax_unreg-Aufruf durchgeführt werden sollen.

Die Schnittstelle für diese Funktion ist:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

Es folgt eine Auflistung der Parameter:

ExitParms (MQAXP) - Ein-/Ausgabe
Parameterstruktur des Exits

ExitContext (MQAXC) - Ein-/Ausgabe
Kontextstruktur des Exits

Rmid (MQLONG) - Ein-/Ausgabe
Ressourcenmanager-ID.

Flags (MQLONG) - Ein-/Ausgabe
Ressourcenmanageroptionen.

XARetCode (MQLONG) - Ein-/Ausgabe
Antwort vom XA-Aufruf.

Aufruf in der Programmiersprache C

Der Warteschlangenmanager definiert logisch die folgenden Variablen:

```
     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 */
```

Dann ruft der Warteschlangenmanager den Exit logisch wie folgt auf:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

Ihr Exit muss dem folgenden Funktionsprototyp in der Programmiersprache C entsprechen:

```
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 */
```

Allgemeine Informationen über das Aufrufen von Exitfunktionen

Dieser Abschnitt stellt eine allgemeine Anleitung zur Verfügung, die Ihnen beim Planen von Exits hilft, insbesondere hinsichtlich beim Umgang mit Fehlern und nicht erwarteten Ereignissen.

Exitfehler

Wenn eine Exitfunktion nach einem zerstörerischen MQGET-Aufruf außerhalb des Synchronisationspunkts abnormal beendet wird, bevor die Nachricht an die Anwendung weitergegeben wurde, kann die Exitverwaltung nach dem Fehler wiederhergestellt werden und der Anwendung die Kontrolle übergeben.

In diesem Fall ist die Nachricht möglicherweise nicht mehr vorhanden. So etwas passiert, wenn eine Anwendung sofort nach dem Erhalt einer Nachricht aus einer Warteschlange fehlschlägt.

Der MQGET-Aufruf schließt möglicherweise mit MQCC_FAILED und MQRC_API_EXIT_ERROR ab.

Wenn eine *Before*-API-Aufruf-Exitfunktion abnormal beendet wird, kann die Exitverwaltung nach dem Fehler wiederhergestellt werden und die Kontrolle an die Anwendung übergeben, ohne den API-Aufruf zu verarbeiten. In solch einem Fall muss die Exitfunktion alle Ressourcen wiederherstellen, deren Eigner sie ist.

Werden verkettete Exits verwendet, können die *After*-API-Aufrufexits für alle *Before*-API-Aufrufexits, die erfolgreich ausgeführt wurden, selbst ausgeführt werden. Der API-Aufruf schlägt möglicherweise mit MQCC_FAILED und MQRC_API_EXIT_ERROR fehl.

Beispiel für Fehlerbehebung für Exitfunktionen

Das folgende Diagramm zeigt die Punkte (e*N*), bei denen Fehler auftreten können. Es ist nur ein Beispiel, das zeigt, wie sich Exits verhalten und sollte zusammen mit der folgenden Tabelle gelesen werden. In diesem Beispiel werden zwei Exitfunktionen vor (before) und nach (after) einem API-Aufruf aufgerufen, um das Verhalten bei verketteten Exits darzustellen.

```
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
```

In der folgenden Tabelle werden die bei jedem Fehlerpunkt erforderlichen Maßnahmen aufgeführt. Es wird nur eine Untergruppe von Fehlerpunkten behandelt, da die angegebenen Regeln auf alle anderen angewandt werden können. Das sind die Maßnahmen, die das für jeden Fall vorgesehene Verhalten angeben.

Tabelle 595. API-Exitfehler und erforderliche geeignete Maßnahmen

Err Pt	**Beschreibung**	**Aktionen**
e1	Fehler beim Einrichten einer Umgebungskonfiguration.	1. Umgebungskonfiguration aufheben, sofern erforderlich 2. Keine Exitfunktionen ausführen 3. Fehler MQCONN mit MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	MQ_INIT_EXIT-Funktion abgeschlossen mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. Umgebung bereinigen 2. Fehler MQCONN mit MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. Umgebung bereinigen
e3	*Before*-Funktion MQ_CONNX_EXIT 1 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. Funktion MQ_TERM_EXIT ausführen 2. Umgebung bereinigen 3. Fehler MQCONN-Aufruf mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. Funktion MQ_TERM_EXIT ausführen, falls erforderlich 3. Umgebung bereinigen, falls erforderlich

Tabelle 595. API-Exitfehler und erforderliche geeignete Maßnahmen (Forts.)

Err Pt	Beschreibung	Aktionen
e4	*Before*-Funktion MQ_CONNX_EXIT 2 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. *After*-Funktion MQ_CONNX_EXIT 1 ausführen 2. Funktion MQ_TERM_EXIT ausführen 3. Umgebung bereinigen 4. Fehler MQCONN-Aufruf mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. *After*-Funktion MQ_CONNX_EXIT 1 ausführen, falls Exit nicht unterdrückt wird 3. Funktion MQ_TERM_EXIT ausführen, falls erforderlich 4. Umgebung bereinigen, falls erforderlich
e5	MQCONN-Aufruf schlägt fehl.	1. MQCONN CompCode und Reason übergeben 2. *After*-Funktion MQ_CONNX_EXIT 2 ausführen, falls *Before*-MQ_CONNX_EXIT 2 erfolgreich war und der Exit nicht unterdrückt wird 3. *After*-Funktion MQ_CONNX_EXIT 1 ausführen, falls *Before*-MQ_CONNX_EXIT 1 erfolgreich war und der Exit nicht unterdrückt wird 4. Funktion MQ_TERM_EXIT ausführen 5. Umgebung bereinigen
e6	*After*-Funktion MQ_CONNX_EXIT 2 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. *After*-Funktion MQ_CONNX_EXIT 1 ausführen 2. MQCONN-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. *After*-Funktion MQ_CONNX_EXIT 1 ausführen, falls erforderlich
e7	*After*-Funktion MQ_CONNX_EXIT 1 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Bei MQXCC_FAILED den MQCONN-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Bei MQXCC_* wie bei den Werten MQXCC_* und MQXR2_* vorgehen[1]
e8	*Before*-Funktion MQ_OPEN_EXIT 1 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Bei MQXCC_FAILED den MQOPEN-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Bei MQXCC_* wie bei den Werten MQXCC_* und MQXR2_* vorgehen[1]

Tabelle 595. API-Exitfehler und erforderliche geeignete Maßnahmen (Forts.)		
Err Pt	**Beschreibung**	**Aktionen**
e9	*Before*-Funktion MQ_OPEN_EXIT 2 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. Die *After*-Funktion MQ_OPEN_EXIT 1 ausführen 2. Den MQOPEN-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Bei MQXCC_* wie bei den Werten MQXCC_* und MQXR2_* vorgehen[1]
e10	MQOPEN-Aufruf schlägt fehl	1. MQOPEN CompCode und Reason übergeben 2. *After*-Funktion MQ_OPEN_EXIT 2 ausführen, falls Exit nicht unterdrückt wird 3. *After*-Funktion MQ_OPEN_EXIT 1 ausführen, falls Exit nicht unterdrückt wird und falls verkettete Exits nicht unterdrückt werden
e11	*After*-Funktion MQ_OPEN_EXIT 2 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. Die *After*-Funktion MQ_OPEN_EXIT 1 ausführen 2. Den MQOPEN-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. *After*-Funktion MQ_OPEN_EXIT 1 ausführen, falls Exit nicht unterdrückt wird
e25	*After*-Funktion MQ_DISC_EXIT 2 schließt ab mit: • MQXCC_FAILED • MQXCC_*	• Für MQXCC_FAILED: 1. *After*-Funktion MQ_DISC_EXIT 1 ausführen 2. Funktion MQ_TERM_EXIT ausführen 3. Exitausführungsumgebung bereinigen 4. MQDISC-Aufruf abschließen mit MQCC_FAILED, MQRC_API_EXIT_ERROR • Für MQXCC_* 1. Vorgehen wie bei den Werten MQXCC_* und MQXR2_*[1] 2. Funktion MQ_TERM_EXIT ausführen 3. Exitausführungsumgebung bereinigen

Anmerkung:

1. Die Werte MQXCC_* und MQXR2_* und ihre jeweiligen Maßnahmen werden in So verarbeiten Warteschlangenmanager Exitfunktionen definiert.

Falsch eingestellte ExitResponse-Felder

In diesem Abschnitt erfahren Sie, was passiert, wenn das ExitResponse-Feld auf Werte eingestellt wird, die nicht unterstützt werden.

Wenn das ExitResponse-Feld auf einen Wert eingestellt wird, der nicht unterstützt wird, sind folgende Aktionen anzuwenden:

• Für eine *Before*-MQCONN- oder MQDISC-API-Exitfunktion:

  - Der ExitResponse2-Wert wird ignoriert.
  - Es werden keine weiteren *Before*-Exitfunktionen in der Exitkette (sofern zutreffend) aufgerufen; der API-Aufruf selbst wird nicht ausgegeben.
  - Bei allen *Before*-Exits, die erfolgreich aufgerufen wurden, werden die *After*-Exits in umgekehrter Reihenfolge aufgerufen.
  - Sofern registriert, werden die Beendigungsexitfunktionen für jene *Before*-MQCONN oder MQDISC-Exitfunktionen in der Kette, die erfolgreich aufgerufen wurden, zu einer Bereinigung nach diesen Exitfunktionen veranlasst.
  - Der MQCONN- oder MQDISC-Aufruf schlägt fehl mit MQRC_API_EXIT_ERROR.
- Bei einer *Before*-WebSphere MQ API-Exitfunktion außer MQCONN oder MQDISC:
  - Der ExitResponse2-Wert wird ignoriert.
  - Es werden keine weiteren *Before*- oder *After*-Datenkonvertierungsfunktionen in der Exitkette (sofern zutreffend) aufgerufen.
  - Bei allen *Before*-Exits, die erfolgreich aufgerufen wurden, werden die *After*-Exits in umgekehrter Reihenfolge aufgerufen.
  - Der WebSphere MQ API-Aufruf selbst wird nicht ausgegeben.
  - Der WebSphere MQ API-Aufruf schlägt mit MQRC_API_EXIT_ERROR fehl.
- Bei einer *After*-MQCONN- oder MQDISC API-Exitfunktion:
  - Der ExitResponse2-Wert wird ignoriert.
  - Die verbleibenden Exitfunktionen, die vor einem API-Aufruf erfolgreich aufgerufen wurden, werden in umgekehrter Reihenfolge aufgerufen.
  - Sofern registriert, werden die Beendigungsexitfunktionen für jene *Before*- oder *After*-MQCONN- oder -MQDISC-Exitfunktionen in der Kette, die erfolgreich aufgerufen wurden, zu einer Bereinigung nach dem Exit veranlasst.
  - Ein CompCode der schwerwiegenderen MQCC_WARNING und der CompCode, der vom Exit an die Anwendung zurückgegeben wird.
  - Ein Reason von MQRC_API_EXIT_ERROR wird an die Anwendung zurückgegeben.
  - Der WebSphere MQ API-Aufruf wurde erfolgreich ausgegeben.
- Für eine *After*-WebSphere MQ API-Aufruf-Exitfunktion, außer MQCONN oder MQDISC:
  - Der ExitResponse2-Wert wird ignoriert.
  - Die verbleibenden Exitfunktionen, die vor einem API-Aufruf erfolgreich aufgerufen wurden, werden in umgekehrter Reihenfolge aufgerufen.
  - Ein CompCode der schwerwiegenderen MQCC_WARNING und der CompCode, der vom Exit an die Anwendung zurückgegeben wird.
  - Ein Reason von MQRC_API_EXIT_ERROR wird an die Anwendung zurückgegeben.
  - Der WebSphere MQ API-Aufruf wurde erfolgreich ausgegeben.
- Für die *Before*-Datenkonvertierung bei der Abrufexitfunktion:
  - Der ExitResponse2-Wert wird ignoriert.
  - Die verbleibenden Exitfunktionen, die vor einem API-Aufruf erfolgreich aufgerufen wurden, werden in umgekehrter Reihenfolge aufgerufen.
  - Die Nachricht wird nicht konvertiert und die nicht konvertierte Nachricht wird an die Anwendung zurückgegeben.
  - Ein CompCode der schwerwiegenderen MQCC_WARNING und der CompCode, der vom Exit an die Anwendung zurückgegeben wird.
  - Ein Reason von MQRC_API_EXIT_ERROR wird an die Anwendung zurückgegeben.
  - Der WebSphere MQ API-Aufruf wurde erfolgreich ausgegeben.

Anmerkung: Da der Fehler beim Exit liegt, ist es besser, MQRC_API_EXIT_ERROR anstatt MQRC_NOT_CONVERTED zurückzugeben.

Wenn eine Exitfunktion das Feld ExitResponse2 auf einen anderen Wert als einen der unterstützten Werte einstellt, wird stattdessen ein Wert von MQXR2_DEFAULT_CONTINUATION vorausgesetzt.

Referenzinformationen zu installierbaren Services

Diese Abschnitte enthalten Referenzinformationen zu den installierbaren Services.

Die Funktionen und Datentypen sind innerhalb der Gruppe für die einzelnen Servicetypen in alphabetischer Reihenfolge aufgeführt.

Anzeige der Funktionen

Dokumentation der Funktionen der installierbaren Services

Für jede Funktion gibt es eine Beschreibung einschließlich der Funktions-ID (für MQZEP).

Die *Parameter* werden in der Reihenfolge aufgelistet, in der sie angegeben werden müssen. Sie müssen alle vorhanden sein.

Hinter jedem Parameternamen steht dessen Datentyp. Dabei handelt es sich um die im Abschnitt „Elementardatentypen” auf Seite 218 beschriebenen Elementardatentypen.

Nach der Beschreibung der Parameter wird auch der Aufruf in der Programmiersprache C beschrieben.

MQZ_AUTHENTICATE_USER - Benutzer authentifizieren

Diese Funktion wird von einer MQZAS_VERSION_5-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager aufgerufen, um einen Benutzer zu authentifizieren oder Identitätskontextfelder festzulegen. Sie wird aufgerufen, wenn der Benutzeranwendungskontext von WebSphere MQ eingerichtet wird.

Der Anwendungskontext wird während Verbindungsaufrufen an dem Punkt der Initialisierung des Benutzerkontexts der Anwendung eingerichtet sowie an jedem Punkt, an dem der Benutzerkontext der Anwendung geändert wird. Bei jedem Verbindungsaufruf werden die Benutzerkontextinformationen der Anwendung im Feld *IdentityContext* erneut abgerufen.

Die Funktions-ID für diese Funktion (für MQZEP) lautet MQZID_AUTHENTICATE_USER.

Syntax

MQZ_AUTHENTICATE_USER (*QMgrName*, *SecurityParms*, *ApplicationContext*, *IdentityContext*, *CorrelationPtr*, *ComponentData*, *Fortsetzung*, *CompCode*, *Ursache*)

Parameter

QMgrName

Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der Name des Warteschlangenmanagers wird zur Information an die Komponente übergeben. Die Berechtigungsserviceschnittstelle erfordert keine definierte Verwendung der Komponente.

SecurityParms

Typ: MQCSP - Eingabe

Sicherheitsparameter Daten, die sich auf die Benutzer-ID, das Kennwort und den Authentifizierungstyp beziehen. Wenn das Attribut AuthenticationType der MQCSP-Struktur als MQCSP_AUTH_USER_ID_AND_PWD angegeben wird, wird sowohl die Benutzer-ID als auch das Kennwort mit den entsprechenden Feldern im Parameter IdentityContext (MQZIC) verglichen, um festzu-

stellen, ob sie übereinstimmen. Weitere Informationen finden Sie in „MQCSP - Sicherheitsparameter” auf Seite 315.

Während des MQI-Aufrufs "MQCONN" enthält dieser Parameter den Wert Null bzw. die Standardwerte.

ApplicationContext
Typ: MQZAC - Eingabe

Anwendungskontext. Daten in Verbindung mit der aufrufenden Anwendung. Einzelheiten dazu finden Sie unter MQZAC - Anwendungskontext.

Bei jedem MQI-Aufruf vom Typ MQCONN oder MQCONNX werden die Benutzerkontextinformationen in der MQZAC-Struktur erneut abgerufen.

IdentityContext
Typ: MQZIC - Eingabe/Ausgabe

Identitätskontext. Bei der Eingabe für die Funktion zum Authentifizieren von Benutzern gibt diese den aktuellen Identitätskontext an. Die Funktion für die Benutzerauthentifizierung kann diesändern. An diesem Punkt übernimmt der Warteschlangenmanager den neuen Identitätskontext. Weitere Details zur MQZIC-Struktur finden Sie unter MQZIC - Identitätskontext.

CorrelationPtr
Typ: MQPTR - Ausgabe

Korrelationsverweis. Gibt die Adresse eventueller Korrelationsdaten an. Dieser Zeiger wird anschließend an andere OAM-Aufrufe übergeben.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager für diese bestimmte Komponente aufbewahrt; alle Änderungen, die von einer der von dieser Komponente bereitgestellten Funktionen an ihr vorgenommen werden, bleiben erhalten und werden beim nächsten Aufruf einer Funktion dieser Komponente's dargestellt.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Continuation
Typ: MQLONG - Ausgabe

Fortsetzungsflag. Sie können folgende Werte angeben:

MQZCI_DEFAULT
Die Fortsetzung hängt von anderen Komponenten ab.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

Weitere Informationen zu diesen Ursachencodes finden Sie unter Ursachencodes.

C-Aufruf

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Deklarieren Sie wie folgt die Parameter, die an den Service übergeben werden:

```
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 - Berechtigung prüfen

Diese Funktion wird von einer MQZAS_VERSION_1-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um zu überprüfen, ob eine Entität über die Berechtigung zum Ausführen einer bestimmten Aktion bzw. mehrerer Aktionen für ein angegebenes Objekt verfügt.

Die Funktions-ID für diese Funktion (für MQZEP) lautet MQZID_CHECK_AUTHORITY.

Syntax

MQZ_CHECK_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityName
Typ: MQCHAR12 - Eingabe

Entitätsname. Der Name der Entität, deren Berechtigung für das Objekt geprüft werden soll. Die maximale Länge der Zeichenfolge beträgt 12 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Die Entität muss dem zugrunde liegenden Sicherheitsservice nicht unbedingt bekannt sein. Ist sie nicht bekannt, werden bei der Prüfung die Berechtigungen der Sondergruppe **nobody** (alle Entitäten

werden als dieser Gruppe zugehörig betrachtet) verwendet. Ein aus Leerzeichen bestehender Name ist gültig und kann auf diese Weise verwendet werden.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der durch "EntityName" angegebene Entitätstyp. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, auf das Zugriff erforderlich ist. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

Berechtigung
Typ: MQLONG - Eingabe

Zu prüfende Berechtigung. Wenn eine Berechtigung geprüft wird, entspricht dieses Feld der jeweiligen Berechtigungsoperation (MQZAO_*-Konstante). Wenn mehrere Berechtigungen geprüft werden, entspricht dies dem bitweisen ODER der betreffenden MQZAO_*-Konstante.

Die folgenden Berechtigungen gelten für die Verwendung des MQI-Aufrufs:

MQZAO_CONNECT
Möglichkeit zur Verwendung des MQCONN-Aufrufs.

MQZAO_BROWSE
Möglichkeit zur Verwendung des MQGET-Aufrufs mit einer Suchoption.

Dies ermöglicht, dass die Option MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR oder MQGMO_BROWSE_NEXT für den MQGET-Aufruf angegeben werden kann.

MQZAO_INPUT
Principal. Möglichkeit zur Verwendung des MQGET-Aufrufs mit einer Eingabeoption.

Dies ermöglicht, dass die Option MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE oder MQOO_INPUT_AS_Q_DEF für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_OUTPUT
Möglichkeit zur Verwendung des MQPUT-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_OUTPUT-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_INQUIRE
Möglichkeit zur Verwendung des MQINQ-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_INQUIRE-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_SET
Möglichkeit zur Verwendung des MQSET-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_SET-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_PASS_IDENTITY_CONTEXT
Fähigkeit zur Übergabe des Identitätskontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_PASS_IDENTITY_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_PASS_IDENTITY_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_PASS_ALL_CONTEXT
Fähigkeit zur Übergabe des gesamten Kontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_PASS_ALL_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_PASS_ALL_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_SET_IDENTITY_CONTEXT
Fähigkeit zur Festlegung des Identitätskontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_SET_IDENTITY_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_SET_IDENTITY_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_SET_ALL_CONTEXT
Fähigkeit zur Festlegung des gesamten Kontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_SET_ALL_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_SET_ALL_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_ALTERNATE_USER_AUTHORITY
Fähigkeit zur Verwendung einer alternativen Benutzerberechtigung.

Diese Berechtigung ermöglicht, dass die Option MQOO_ALTERNATE_USER_AUTHORITY für den MQOPEN-Aufruf und die Option MQPMO_ALTERNATE_USER_AUTHORITY für den MQPUT1-Aufruf angegeben werden kann.

MQZAO_ALL_MQI
Alle MQI-Berechtigungen.

Dies aktiviert alle Berechtigungen.

Für die Verwaltung eines Warteschlangenmanagers gelten die folgenden Berechtigungen:

MQZAO_CREATE
Möglichkeit zur Erstellung von Objekten eines angegebenen Typs.

MQZAO_DELETE
Möglichkeit zum Löschen eines angegebenen Objekts.

MQZAO_DISPLAY
Möglichkeit zum Anzeigen der Attribute eines angegebenen Objekts.

MQZAO_ÄNDERUNG
Möglichkeit zum Ändern der Attribute eines angegebenen Objekts.

MQZAO_CLEAR
Möglichkeit zum Löschen aller Nachrichten aus einer angegebenen Warteschlange.

MQZAO_AUTHORIZE
Möglichkeit zur Autorisierung anderer Benutzer für ein angegebenes Objekt.

MQZAO_CONTROL
Möglichkeit, ein Empfangsprogramm, einen Service oder ein Kanalobjekt, das sich nicht auf einem Client befindet, zu starten oder zu stoppen, und die Möglichkeit, ein Kanalobjekt, das sich nicht auf einem Client befindet, mit Ping zu überprüfen.

MQZAO_CONTROL_EXTENDED
Möglichkeit zum Zurücksetzen einer Folgenummer bzw. zum Auflösen einer unbestätigten Nachricht in einem Kanalobjekt, das sich nicht auf dem Client befindet.

MQZAO_ALL_ADMIN
Fähigkeit zur Festlegung des Identitätskontexts.

Alle Administrationsberechtigungen mit Ausnahme von MQZAO_CREATE.

Die folgenden Berechtigungen gelten sowohl für die Verwendung des MQI als auch für die Administration eines Warteschlangenmanagers:

MQZAO_ALL
Alle Berechtigungen mit Ausnahme von MQZAO_CREATE.

MQZAO_NONE
Keine Berechtigungen.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

Wenn der Aufruf einer Komponente fehlschlägt (das heißt, *CompCode* gibt MQCC_FAILED zurück) und der Parameter *Continuation* den Wert MQZCI_DEFAULT oder MQZCI_CONTINUE hat, ruft der Warteschlangenmanager weitere Komponenten auf, sofern welche vorhanden sind.

Wenn der Aufruf erfolgreich ist (das heißt, wenn *CompCode* den Wert MQCC_OK zurückgibt), werden keine anderen Komponenten aufgerufen, unabhängig von der Einstellung von *Continuation*.

Wenn der Aufruf fehlschlägt und der Parameter *Continuation* den Wert MQZCI_STOP hat, werden keine weiteren Komponenten aufgerufen und der Fehler wird an den Warteschlangenmanager zurückgegeben. Komponenten verfügen über keine Informationen über vorherige Aufrufe, folglich ist der Parameter *Continuation* vor dem Aufruf stets auf MQZCI_DEFAULT gesetzt.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigung prüfen (erweitert)

Diese Funktion wird von einer MQZAS_VERSION_2-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um zu überprüfen, ob eine Entität über die Berechtigung zum Ausführen einer bestimmten Aktion bzw. mehrerer Aktionen für ein angegebenes Objekt verfügt.

Die Funktions-ID für diese Funktion (für MQZEP) lautet MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 entspricht MQZ_CHECK_AUTHORITY, mit dem einzigen Unterschied, dass der Parameter *EntityName* durch den Parameter *EntityData* ersetzt ist.

Syntax

MQZ_CHECK_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityData
Typ: MQZED - Eingabe

Entitätsdaten. Daten, die sich auf die Entität mit Berechtigung für das zu prüfende Objekt beziehen. Weitere Informationen finden Sie im Abschnitt „MQZED - Entitätsdeskriptor" auf Seite 1237.

Die Entität muss dem zugrunde liegenden Sicherheitsservice nicht unbedingt bekannt sein. Ist sie nicht bekannt, werden bei der Prüfung die Berechtigungen der Sondergruppe **nobody** (alle Entitäten werden als dieser Gruppe zugehörig betrachtet) verwendet. Ein aus Leerzeichen bestehender Name ist gültig und kann auf diese Weise verwendet werden.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der Typ der durch *EntityData* angegebenen Entität. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, auf das Zugriff erforderlich ist. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Zu prüfende Berechtigung. Wenn eine Berechtigung geprüft wird, entspricht dieses Feld der jeweiligen Berechtigungsoperation (MQZAO_*-Konstante). Wenn mehrere Berechtigungen geprüft werden, entspricht dies dem bitweisen ODER der betreffenden MQZAO_*-Konstante.

Die folgenden Berechtigungen gelten für die Verwendung des MQI-Aufrufs:

MQZAO_CONNECT
Möglichkeit zur Verwendung des MQCONN-Aufrufs.

MQZAO_BROWSE
Möglichkeit zur Verwendung des MQGET-Aufrufs mit einer Suchoption.

Dies ermöglicht, dass die Option MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR oder MQGMO_BROWSE_NEXT für den MQGET-Aufruf angegeben werden kann.

MQZAO_INPUT
Principal. Möglichkeit zur Verwendung des MQGET-Aufrufs mit einer Eingabeoption.

Dies ermöglicht, dass die Option MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE oder MQOO_INPUT_AS_Q_DEF für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_OUTPUT
Möglichkeit zur Verwendung des MQPUT-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_OUTPUT-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_INQUIRE
Möglichkeit zur Verwendung des MQINQ-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_INQUIRE-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_SET
Möglichkeit zur Verwendung des MQSET-Aufrufs.

Diese Berechtigung ermöglicht, dass die MQOO_SET-Option für den MQOPEN-Aufruf angegeben werden kann.

MQZAO_PASS_IDENTITY_CONTEXT
Fähigkeit zur Übergabe des Identitätskontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_PASS_IDENTITY_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_PASS_IDENTITY_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_PASS_ALL_CONTEXT
Fähigkeit zur Übergabe des gesamten Kontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_PASS_ALL_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_PASS_ALL_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_SET_IDENTITY_CONTEXT
Fähigkeit zur Festlegung des Identitätskontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_SET_IDENTITY_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_SET_IDENTITY_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_SET_ALL_CONTEXT
Fähigkeit zur Festlegung des gesamten Kontexts.

Diese Berechtigung ermöglicht, dass die Option MQOO_SET_ALL_CONTEXT für den MQOPEN-Aufruf und die Option MQPMO_SET_ALL_CONTEXT für MQPUT- und MQPUT1-Aufrufe angegeben werden kann.

MQZAO_ALTERNATE_USER_AUTHORITY
Fähigkeit zur Verwendung einer alternativen Benutzerberechtigung.

Diese Berechtigung ermöglicht, dass die Option MQOO_ALTERNATE_USER_AUTHORITY für den MQOPEN-Aufruf und die Option MQPMO_ALTERNATE_USER_AUTHORITY für den MQPUT1-Aufruf angegeben werden kann.

MQZAO_ALL_MQI
Alle MQI-Berechtigungen.

Dies aktiviert alle Berechtigungen.

Für die Verwaltung eines Warteschlangenmanagers gelten die folgenden Berechtigungen:

MQZAO_CREATE
Möglichkeit zur Erstellung von Objekten eines angegebenen Typs.

MQZAO_DELETE
Möglichkeit zum Löschen eines angegebenen Objekts.

MQZAO_DISPLAY
Möglichkeit zum Anzeigen der Attribute eines angegebenen Objekts.

MQZAO_ÄNDERUNG
Möglichkeit zum Ändern der Attribute eines angegebenen Objekts.

MQZAO_CLEAR
Möglichkeit zum Löschen aller Nachrichten aus einer angegebenen Warteschlange.

MQZAO_AUTHORIZE
Möglichkeit zur Autorisierung anderer Benutzer für ein angegebenes Objekt.

MQZAO_CONTROL
Möglichkeit, ein Empfangsprogramm, einen Service oder ein Kanalobjekt, das sich nicht auf einem Client befindet, zu starten oder zu stoppen, und die Möglichkeit, ein Kanalobjekt, das sich nicht auf einem Client befindet, mit Ping zu überprüfen.

MQZAO_CONTROL_EXTENDED
Möglichkeit zum Zurücksetzen einer Folgenummer bzw. zum Auflösen einer unbestätigten Nachricht in einem Kanalobjekt, das sich nicht auf dem Client befindet.

MQZAO_ALL_ADMIN
Fähigkeit zur Festlegung des Identitätskontexts.

Alle Administrationsberechtigungen mit Ausnahme von MQZAO_CREATE.

Die folgenden Berechtigungen gelten sowohl für die Verwendung des MQI als auch für die Administration eines Warteschlangenmanagers:

MQZAO_ALL
Alle Berechtigungen mit Ausnahme von MQZAO_CREATE.

MQZAO_NONE
Keine Berechtigungen.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Prüfung, ob Benutzer privilegiert ist

Diese Funktion wird von einer MQZAS_VERSION_6-Berechtigungsservicekomponente bereitgestellt. Sie wird vom Warteschlangenmanager aufgerufen, um zu bestimmen, ob es sich bei einem angegebenen Benutzer um einen privilegierten Benutzer handelt.

Die Funktions-ID für diese Funktion (für MQZEP) ist MQZID_CHECK_PRIVILEGED.

Syntax

MQZ_CHECK_PRIVILEGED(*QMgrName*, *EntityData*, *EntityType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityData
Typ: MQZED - Eingabe

Entitätsdaten. Daten zur Entität, die geprüft werden soll. Weitere Informationen finden Sie unter „MQZED - Entitätsdeskriptor” auf Seite 1237.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der Typ der von EntityData angegebenen Entität. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

Wenn der Aufruf einer Komponente fehlschlägt (das heißt, *CompCode* gibt MQCC_FAILED zurück) und der Parameter *Continuation* den Wert MQZCI_DEFAULT oder MQZCI_CONTINUE hat, ruft der Warteschlangenmanager weitere Komponenten auf, sofern welche vorhanden sind.

Wenn der Aufruf erfolgreich ist (das heißt, wenn *CompCode* den Wert MQCC_OK zurückgibt), werden keine anderen Komponenten aufgerufen, unabhängig von der Einstellung von *Continuation*.

Wenn der Aufruf fehlschlägt und der Parameter *Continuation* den Wert MQZCI_STOP hat, werden keine weiteren Komponenten aufgerufen und der Fehler wird an den Warteschlangenmanager zurückgegeben. Komponenten verfügen über keine Informationen über vorherige Aufrufe, folglich ist der Parameter *Continuation* vor dem Aufruf stets auf MQZCI_DEFAULT gesetzt.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_PRIVILEGED
(2584, X'A18') Dieser Benutzer verfügt nicht über die ID eines privilegierten Benutzers.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie im Abschnitt API-Ursachencode.

C-Aufruf

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Alle Berechtigungen kopieren

Diese Funktion wird von einer Berechtigungsservicekomponente bereitgestellt. Sie wird vom Warteschlangenmanager gestartet, um alle Berechtigungen zu kopieren, die aktuell für ein Referenzobjekt für ein anderes Objekt wirksam sind.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_COPY_ALL_AUTHORITY.

Syntax

MQZ_COPY_ALL_AUTHORITY(*QMgrName*, *RefObjectName*, *ObjectName*, *ObjectType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

RefObjectName
Typ: MQCHAR48 - Eingabe

Name des Referenzobjekts. Der Name des Referenzobjekts, für das die Berechtigungen kopiert werden sollen. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das der Zugriff festgelegt werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *RefObjectName* und *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Referenzobjekt ist unbekannt.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigung löschen

Diese Funktion wird von einer Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um alle dem angegebenen Objekt zugeordneten Berechtigungen zu löschen.

Die Funktions-ID für diese Funktion (für MQZEP) ist MQZID_DELETE_AUTHORITY.

Syntax

MQZ_DELETE_AUTHORITY(*QMgrName*, *ObjectName*, *ObjectType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das der Zugriff gelöscht werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA - Berechtigungsdaten aufzählen

Diese Funktion wird von einer MQZAS_VERSION_4-Berechtigungsservicekomponente bereitgestellt und wiederholt vom Warteschlangenmanager gestartet, um alle Berechtigungsdaten abzurufen, die den für den ersten Aufruf angegebenen Auswahlkriterien entsprechen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_ENUMERATE_AUTHORITY_DATA.

Syntax

MQZ_ENUMERATE_AUTHORITY_DATA(*QMgrName*, *StartEnumeration*, *Filter*, *AuthorityBufferLength*, *AuthorityBuffer*, *AuthorityDataLength*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

StartEnumeration
Typ: MQLONG - Eingabe

Flag, das anzeigt, ob ein Aufruf die Aufzählung starten kann. Dieses zeigt an, ob der Aufruf die Aufzählung von Berechtigungsdaten starten bzw. die Aufzählung von Berechtigungsdaten fortsetzen kann, die durch einen vorherigen Aufruf von MQZ_ENUMERATE_AUTHORITY_DATA gestartet wurde. Folgende Werte sind möglich:

MQZSE_START
Aufzählung starten. Der Aufruf wird mit diesem Wert gestartet, um die Aufzählung der Berechtigungsdaten zu starten. Der Parameter *Filter* legt die Auswahlkriterien fest, die bei der Auswahl der bei diesem und nachfolgenden Aufrufen zurückgegebenen Berechtigungsdaten gelten sollen.

MQZSE_CONTINUE
Aufzählung fortsetzen. Der Aufruf wird mit diesem Wert gestartet, um die Aufzählung der Berechtigungsdaten fortzusetzen. Der Parameter *Filter* wird in diesem Fall ignoriert und kann als Nullzeiger festgelegt werden (die Auswahlkriterien werden durch den für den Aufruf geltenden Parameter *Filter* angegeben, für den der Parameter *StartEnumeration* auf MQZSE_START gesetzt war).

Filter
Typ: MQZAD - Eingabe

Filter. Wenn der Parameter *StartEnumeration* den Wert MQZSE_START hat, legt *Filter* die Auswahlkriterien fest, die bei der Auswahl der zurückzugebenen Berechtigungsdaten gelten sollen. Wenn *Filter* der Nullzeiger ist, gelten keine Auswahlkriterien, das heißt, es werden alle Berechtigungsdaten zurückgegeben. Ausführliche Informationen zu den möglichen Auswahlkriterien finden Sie in „MQZAD - Berechtigungsdaten” auf Seite 1234.

Wenn *StartEnumeration* den Wert MQZSE_CONTINUE hat, wird der Parameter *Filter* ignoriert und kann als Nullzeiger festgelegt werden.

AuthorityBufferLength
Typ: MQLONG - Eingabe

Die Länge von *AuthorityBuffer*. Dies ist die Länge des Parameters *AuthorityBuffer* in Byte. Der Berechtigungspuffer muss groß genug sein, um die zurückzugebenden Daten aufnehmen zu können.

AuthorityBuffer
Typ: MQZAD - Ausgabe

Berechtigungsdaten. Dies ist der Puffer, in dem die Berechtigungsdaten zurückgegeben werden. Der Puffer muss groß genug sein, um eine MQZAD-Struktur, eine MQZED-Struktur sowie den längsten definierten Entitätsnamen und den längsten definierten Domänennamen aufnehmen zu können.

Anmerkung: Hinweis: Dieser Parameter ist als MQZAD definiert, da der Typ MQZAD immer zum Start des Puffers auftritt. Ein als MQZAD deklarierter Puffer wird jedoch nicht die erforderliche Größe haben. Er muss größer als ein Parameter vom Typ MQZAD sein, sodass er MQZAD, MQZED sowie Entitäts- und Domänennamen aufnehmen kann.

AuthorityDataLength
Typ: MQLONG - Ausgabe

Länge der in *AuthorityBuffer* zurückgegebenen Daten. Wenn der Berechtigungspuffer zu klein ist, wird *AuthorityDataLength* auf die erforderliche Pufferlänge gesetzt und der Aufruf gibt den Beendigungscode MQCC_FAILED und den Ursachencode MQRC_BUFFER_LENGTH_ERROR zurück.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_ENUMERATE_AUTHORITY_DATA hat dies die gleichen Auswirkungen wie MQZCI_CONTINUE.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Parameter für Puffergröße ist nicht gültig.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') Keine Daten verfügbar.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Benutzer freigeben

Diese Funktion wird von einer MQZAS_VERSION_5-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um zugehörige zugeordnete Ressourcen freizugeben.

Sie wird gestartet, wenn die Ausführung einer Anwendung unter allen Benutzerkontexten abgeschlossen wurde, beispielsweise bei einem MQDISC MQI-Aufruf.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_FREE_USER.

Syntax

MQZ_FREE_USER(*QMgrName*, *FreeParms*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName

Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

FreeParms

Typ: MQZFP - Eingabe

Freie Parameter. Eine Struktur, die Daten bezüglich der freizugebenden Ressource enthält. Weitere Informationen finden Sie im Artikel „MQZFP - Parameter freigeben” auf Seite 1239.

ComponentData

Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Continuation

Typ: MQLONG - Ausgabe

Fortsetzungsflag. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt von anderen Komponenten ab.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigung abrufen

Diese Funktion wird von einer MQZAS_VERSION_1-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung einer Entität für den Zugriff auf das angegebene Objekt abzurufen. Wenn die Entität ein Principal ist, gehören hierzu auch die Berechtigungen der Gruppen, deren Mitglied der Principal ist. Ebenso umfasst die Rückgabe die Berechtigungen generischer Profile.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_GET_AUTHORITY.

Syntax

MQZ_GET_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityName
Typ: MQCHAR12 - Eingabe

Entitätsname. Der Name der Entität, deren Zugriff auf das Objekt abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 12 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der über *EntityName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das der Zugriff abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der über *ObjectName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn die Entität über eine Berechtigung verfügt, entspricht dieses Feld der jeweiligen Berechtigungsoperation (Konstante MQZAO_*). Wenn sie über mehrere Berechtigungen verfügt, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_GET_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_CONTINUE.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie im Abschnitt API-Ursachencode.

C-Aufruf

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigung abrufen (erweitert)

Diese Funktion wird von einer MQZAS_VERSION_2-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung einer Entität für den Zugriff auf das angegebene Objekt abzurufen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 entspricht MQZ_GET_AUTHORITY, mit dem einzigen Unterschied, dass der Parameter *EntityName* durch den Parameter *EntityData* ersetzt ist.

Syntax

MQZ_GET_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityData
Typ: MQZED - Eingabe

Entitätsdaten. Daten, die sich auf die Entität beziehen, für die die Berechtigung für das Objekt abgerufen werden soll. Weitere Informationen finden Sie im Abschnitt „MQZED - Entitätsdeskriptor" auf Seite 1237.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der Typ der durch *EntityData* angegebenen Entität. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das die Entitätsberechtigung abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn die Entität über eine Berechtigung verfügt, entspricht dieses Feld der jeweiligen Berechtigungsoperation (Konstante MQZAO_*). Wenn sie über mehrere Berechtigungen verfügt, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

Syntax

MQZ_GET_AUTHORITY_2*(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C-Aufruf

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Explizite Berechtigung abrufen

Diese Funktion wird von einer MQZAS_VERSION_1-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung abzurufen, über die eine benannte Gruppe verfügen muss, um auf ein angegebenes Objekt zugreifen zu können (jedoch ohne die zusätzliche Berechtigung der Gruppe **nobody**), bzw. die Berechtigung, über die die Primärgruppe des benannten Principals verfügt, um auf ein angegebenes Objekt zuzugreifen.

Unter UNIX wird für den integrierten WebSphere MQ-Objektberechtigungsmanager nur die Berechtigung der Primärgruppe des Principals zurückgegeben.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_GET_EXPLICIT_AUTHORITY.

Syntax

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityName
Typ: MQCHAR12 - Eingabe

Entitätsname. Der Name der Entität, für die der Zugriff auf das Objekt abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 12 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der über *EntityName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das die Entitätsberechtigung abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der über *ObjectName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn die Entität über eine Berechtigung verfügt, entspricht dieses Feld der jeweiligen Berechtigungsoperation (Konstante MQZAO_*). Wenn sie über mehrere Berechtigungen verfügt, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_GET_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_CONTINUE.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie im Abschnitt API-Ursachencode.

C-Aufruf

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Explizite Berechtigung abrufen (erweitert)

Diese Funktion wird von einer MQZAS_VERSION_2-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung abzurufen, über die eine benannte Gruppe verfügen muss, um auf ein angegebenes Objekt zugreifen zu können (jedoch ohne die zusätzliche Berechtigung der Gruppe **nobody**), bzw. die Berechtigung, über die die Primärgruppe des benannten Principals verfügt, um auf ein angegebenes Objekt zuzugreifen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 entspricht MQZ_GET_EXPLICIT_AUTHORITY, mit dem einzigen Unterschied, dass der Parameter *EntityName* durch den Parameter *EntityData* ersetzt ist.

Syntax

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityData
Typ: MQZED - Eingabe

Entitätsdaten. Daten, die sich auf die Entität beziehen, deren Berechtigung für das Objekt abgerufen werden soll. Weitere Informationen finden Sie im Abschnitt „MQZED - Entitätsdeskriptor" auf Seite 1237.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der Typ der durch *EntityData* angegebenen Entität. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das die Entitätsberechtigung abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn die Entität über eine Berechtigung verfügt, entspricht dieses Feld der jeweiligen Berechtigungsoperation (Konstante MQZAO_*). Wenn sie über mehrere Berechtigungen verfügt, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigungsservice initialisieren

Diese Funktion wird von einer Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager bei der Konfiguration der Komponente gestartet. Es wird erwartet, dass sie MQZEP aufruft, um dem Warteschlangenmanager Informationen bereitzustellen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_INIT_AUTHORITY.

Syntax

MQZ_INIT_AUTHORITY(*Hconfig*, *Options*, *QMgrName*, *ComponentDataLength*, *ComponentData*, *Version*, *CompCode*, *Reason*)

Parameter

Hconfig
Typ: MQHCONFIG - Eingabe

Konfigurationskennung. Diese Kennung stellt die spezielle Komponente dar, die gerade initialisiert wird. Sie muss von der Komponente beim Aufruf des Warteschlangenmanagers mit der MQZEP-Funktion verwendet werden.

Optionen
Typ: MQLONG - Eingabe

Initialisierungsoptionen. Folgende Werte sind zulässig:

MQZIO_PRIMARY
Primärinitialisierung.

MQZIO_SECONDARY
Sekundärinitialisierung.

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ComponentDataLength
Typ: MQLONG - Eingabe

Länge der Komponentendaten. Länge des Bereichs *ComponentData* in Byte. Diese Länge wird in den Daten der Komponentenkonfiguration definiert.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Dieser Parameter wird ausschließlich mit Nullen initialisiert, bevor die primäre Initialisierungsfunktion der Komponente aufgerufen wird. Der Warteschlangenmanager speichert diese Daten für diese spezielle Komponente. Alle Änderungen, die die von der Komponente bereitgestellten Funktionen (einschließlich der Initialisierungsfunktion) daran vornehmen, werden aufgezeichnet und beim nächsten Aufruf einer dieser Komponentenfunktionen angezeigt.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Version
Typ: MQLONG - Eingabe/Ausgabe

Versionsnummer. Bei der Eingabe für die Initialisierungsfunktion gibt dieser Wert die höchste Versionsnummer an, die der Warteschlangenmanager unterstützt. Die Initialisierungsfunktion muss diesen Wert gegebenenfalls in die Nummer der von ihr unterstützten Schnittstellenversion ändern. Wenn der Warteschlangenmanager die von der Komponente zurückgegebene Version nicht unterstützt, ruft er die Komponentenfunktion MQZ_TERM_AUTHORITY auf und verwendet diese Komponente nicht weiter.

Folgende Werte werden unterstützt:

MQZAS_VERSION_1
Version 1.

MQZAS_VERSION_2
Version 2.

MQZAS_VERSION_3
Version 3.

MQZAS_VERSION_4
Version 4.

MQZAS_VERSION_5
Version 5.

MQZAS_VERSION_6
Version 6.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Initialisierung aus nicht definiertem Grund fehlgeschlagen.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigungsservice abfragen

Diese Funktion wird von einer MQZAS_VERSION_5-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die unterstützten Funktionen abzurufen.

Wenn mehrere Servicekomponenten in Verwendung sind, werden diese in umgekehrter Reihenfolge aufgerufen, in der sie installiert wurden.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_INQUIRE.

Syntax

MQZ_INQUIRE(*QMgrName*, *SelectorCount* , *Selectors* , *IntAttrCount* , *IntAttrs* , *CharAttrLength* , *CharAttrs* , *SelectorReturned* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

SelectorCount
Typ: MQLONG - Eingabe

Anzahl der Selektoren. Die Anzahl der im Parameter *Selectors* bereitgestellten Selektoren.

Der Wert muss im Bereich von 0 bis 256 liegen.

Selectors
Typ: MQLONGxSelectorCount - Eingabe

Feldgruppe der Selektoren. Jeder Selektor identifiziert ein erforderliches Attribut und muss einer der folgenden sein:

- MQIACF_INTERFACE_VERSION (Ganzzahl)
- MQIACF_USER_ID_SUPPORT (Ganzzahl)
- MQCACF_SERVICE_COMPONENT (Zeichen)

Selektoren können in beliebiger Reihenfolge angegeben werden. Die Anzahl der Selektoren in der Feldgruppe wird durch den Parameter *SelectorCount* angegeben.

Ganzzahlattribute, die durch Selektoren angegeben werden, werden im Parameter *IntAttrs* in derselben Reihenfolge zurückgegeben, in der sie in *Selectors* erscheinen.

Zeichenattribute, die durch Selektoren angegeben werden, werden im Parameter *CharAttrs* in derselben Reihenfolge zurückgegeben, in der sie angezeigt werden *Selectors* .

IntAttrCount
Typ: MQLONG - Eingabe

Anzahl der im Parameter "IntAttrs" bereitgestellten Ganzzahlattribute.

Der Wert muss im Bereich von 0 bis 256 liegen.

IntAttrs
Typ: MQLONG×IntAttrCount - Ausgabe

Ganzzahlattribute. Feldgruppe der Ganzzahlattribute. Die Ganzzahlenattribute werden in derselben Reihenfolge wie die entsprechenden Ganzzahlselektoren im Array *Selectors* zurückgegeben.

CharAttrCount
Typ: MQLONG - Eingabe

Länge des Zeichenattributpuffers. Die Länge des Parameters *CharAttrs* in Byte.

Der Wert muss mindestens der Summe der Längen der angeforderten Zeichenattribute entsprechen. Wenn keine Zeichenattribute angefordert werden, ist Null als Wert zulässig.

CharAttrs
Typ: MQLONG×CharAttrCount - Ausgabe

Puffer für Zeichenattribute. Puffer, der miteinander verkettete Zeichenattribute enthält. Die Zeichenattribute werden in derselben Reihenfolge wie die entsprechenden Zeichenselektoren im Array *Selectors* zurückgegeben.

Die Länge des Puffers ist im Parameter "CharAttrCount" festgelegt.

SelectorReturned
Typ: MQLONG×SelectorCount - Eingabe

Zurückgegebener Selektor. Feldgruppe von Werten, die angeben, welche Attribute bereits aus der Gruppe zurückgegeben wurden, die von den Selektoren im Parameter "Selectors" angefordert wurde. Die Anzahl der Werte in diesem Array wird durch den Parameter *SelectorCount* angegeben. Jeder Wert in der Feldgruppe bezieht sich auf den Selektor an der entsprechenden Position in der Feldgruppe "Selectors". Folgende Werte sind möglich:

MQZSL_RETURNED
Das vom entsprechenden Selektor im Parameter *Selectors* angeforderte Attribut wurde zurückgegeben.

MQZSL_NOT_RETURNED
Das vom entsprechenden Selektor im Parameter *Selectors* angeforderte Attribut wurde nicht zurückgegeben.

Das Array wird mit allen Werten als *MQZSL_NOT_RETURNED*initialisiert. Wenn eine Berechtigungsservicekomponente ein Attribut zurückgibt, setzt sie den entsprechenden Wert im Array auf

MQZSL_NOT_RETURNED . Dies ermöglicht jeder beliebigen anderen Berechtigungsservicekomponente, an die der Aufruf "Inquire" erfolgt, festzustellen, welche Attribute bereits zurückgegeben wurden.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übergeben.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Teilweise Beendigung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

CompCode mit Ursachencode.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_CHAR_ATTRS_TOO_SHORT
Unzureichender Speicherplatz für Zeichenattribute

MQRC_INT_COUNT_TOO_SMALL
Unzureichender Speicherplatz für Ganzzahlattribute

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SELECTOR_COUNT_ERROR
Die Anzahl der Selektoren ist ungültig.

MQRC_SELECTOR_ERROR
Der Attributselektor ist ungültig.

MQRC_SELECTOR_LIMIT_EXCEEDED
Zu viele Selektoren angegeben.

MQRC_INT_ATTR_COUNT_ERROR
Die Anzahl der Ganzzahlattribute ist ungültig.

MQRC_INT_ATTRS_ARRAY_ERROR
Das Ganzzahlattributarray ist ungültig.

MQRC_CHAR_ATTR_LENGTH_ERROR
Die Anzahl der Zeichenattribute ist ungültig.

MQRC_CHAR_ATTRS_ERROR
Die Zeichenattributzeichenfolge ist ungültig.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Alle Berechtigungen aktualisieren

Diese Funktion wird von einer MQZAS_VERSION_3-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager aufgerufen, um die interne Berechtigungsliste der Komponente zu aktualisieren.

Die Funktions-ID für diese Funktion (für MQZEP) lautet MQZID_REFRESH_CACHE (8L).

Syntax

MQZ_REFRESH_CACHE(*QMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente beibehalten. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen bleiben erhalten und werden beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

C-Aufruf

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
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 - Berechtigung festlegen

Diese Funktion wird von einer MQZAS_VERSION_1-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung einer Entität für den Zugriff auf das angegebene Objekt festzulegen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_SET_AUTHORITY.

Anmerkung: Diese Funktion überschreibt bereits vorhandene Berechtigungen. Sollen bereits vorhandene Berechtigungen beibehalten werden, müssen sie mit dieser Funktion erneut festgelegt werden.

Syntax

MQZ_SET_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityName
Typ: MQCHAR12 - Eingabe

Entitätsname. Der Name der Entität, für die der Zugriff auf das Objekt abgerufen werden soll. Die maximale Länge der Zeichenfolge beträgt 12 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der über *EntityName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, auf das Zugriff erforderlich ist. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der über *ObjectName* angegebene Entitätstyp. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn eine Berechtigung festgelegt wird, entspricht der Wert in diesem Feld der betreffenden Berechtigungsoperation (Konstante "MQZAO_*"). Wenn mehrere Berechtigungen festgelegt werden, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_GET_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_CONTINUE.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigung festlegen (erweitert)

Diese Funktion wird von einer MQZAS_VERSION_2-Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um die Berechtigung einer Entität für den Zugriff auf das angegebene Objekt festzulegen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_SET_AUTHORITY.

Anmerkung: Diese Funktion überschreibt bereits vorhandene Berechtigungen. Sollen bereits vorhandene Berechtigungen beibehalten werden, müssen sie mit dieser Funktion erneut festgelegt werden.

MQZ_SET_AUTHORITY_2 entspricht MQZ_SET_AUTHORITY, mit dem einzigen Unterschied, dass der Parameter *EntityName* durch den Parameter *EntityData* ersetzt ist.

Syntax

MQZ_SET_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

EntityData
Typ: MQZED - Eingabe

Entitätsdaten. Daten, die sich auf die Entität beziehen, deren Berechtigung für das Objekt festgelegt werden soll. Weitere Informationen finden Sie im Abschnitt „MQZED - Entitätsdeskriptor" auf Seite 1237.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp. Der Typ der durch *EntityData* angegebenen Entität. Folgende Werte sind zulässig:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Gruppe.

ObjectName
Typ: MQCHAR48 - Eingabe

Objektname Der Name des Objekts, für das die Entitätsberechtigung festgelegt werden soll. Die maximale Länge der Zeichenfolge beträgt 48 Zeichen. Kürzere Zeichenfolgen werden nach rechts durch Leerzeichen aufgefüllt. Der Name wird nicht durch ein Nullzeichen beendet.

Wenn *ObjectType* den Wert "MQOT_Q_MGR" hat, entspricht dieser Name *QMgrName*.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp. Der Typ der durch *ObjectName* angegebenen Entität. Folgende Werte sind zulässig:

MQOT_AUTH_INFO
Authentifizierungsdaten.

MQOT_CHANNEL
Der Kanal.

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal.

MQOT_LISTENER
Empfangsprogramm.

MQOT_NAMELIST
Namensliste.

MQOT_PROCESS
Prozessdefinition.

MQOT_Q
Queue.

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service.

MQOT_TOPIC
Thema.

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung der Entität. Wenn eine Berechtigung festgelegt wird, entspricht der Wert in diesem Feld der betreffenden Berechtigungsoperation (Konstante "MQZAO_*"). Wenn mehrere Berechtigungen festgelegt werden, entspricht dies dem bitweisen ODER der entsprechenden MQZAO_*-Konstante.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte können angegeben werden:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

Für MQZ_CHECK_AUTHORITY hat dies die gleichen Auswirkungen wie MQZCI_STOP.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Keine Zugriffsberechtigung.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Dem Service unbekannte Entität.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Berechtigungsservice beenden

Diese Funktion wird von einer Berechtigungsservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, wenn dieser die Services einer Komponente nicht mehr benötigt. Von dieser Funktion müssen alle für die Komponente erforderlichen Bereinigungsmaßnahmen durchgeführt werden.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_TERM_AUTHORITY.

Syntax

MQZ_TERM_AUTHORITY(*Hconfig*, *Options*, *QMgrName*, *ComponentData*, *CompCode*, *Reason*)

Parameter

Hconfig
Typ: MQHCONFIG - Eingabe

Konfigurationskennung. Diese Kennung stellt die Komponente dar, die beendet wird. Sie muss von der Komponente beim Aufruf des Warteschlangenmanagers mit der MQZEP-Funktion verwendet werden.

Optionen
Typ: MQLONG - Eingabe

Beendigungsoptionen. Folgende Werte sind zulässig:

MQZTO_PRIMARY
Primärbeendigung.

MQZTO_SECONDARY
Sekundäre Beendigung.

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter "ComponentDataLength" des MQZ_INIT_AUTHORITY-Aufrufs übergeben.

Nachdem der MQZ_TERM_AUTHORITY-Aufruf abgeschlossen wurde, verwirft der Warteschlangenmanager diese Daten.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_TERMINATION_FAILED
(2287, X'8FF') Beendigung aus nicht definiertem Grund fehlgeschlagen.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Name löschen

Diese Funktion wird von einer Namensservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um einen Eintrag der angegebenen Warteschlange zu löschen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_DELETE_NAME.

Syntax

MQZ_DELETE_NAME(*QMgrName*, *QName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

Warteschlangenname
Typ: MQCHAR48 - Eingabe

Der Name der Warteschlange. Der Name der Warteschlange, aus der ein Eintrag gelöscht werden soll. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Diese Daten werden vom Warteschlangenmanager im Namen dieser bestimmten Komponente aufbewahrt. Jegliche Änderungen an diesen Daten durch eine der von dieser Komponente bereitgestellten Funktionen werden übernommen und beim nächsten Aufruf einer der Funktionen dieser Komponente vorgeschlagen.

Der Warteschlangenmanager übergibt die Länge dieses Datenbereichs an den Parameter ComponentDataLength des Aufrufs von MQZ_INIT_NAME.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Folgende Werte sind zulässig:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

Bei dem Befehl **MQZ_DELETE_NAME** versucht der Warteschlangenmanager nicht, eine andere Komponente zu starten, unabhängig vom Rückgabewert des Parameters **Continuation**.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_WARNING
Warnung (teilweise Ausführung)

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_WARNING gesetzt ist:

MQRC_UNKNOWN_NAME
(2288, X'8F0') Name der Warteschlange wurde nicht gefunden.

Anmerkung: Möglicherweise kann dieser Code nicht zurückgegeben werden, wenn der zugrunde liegende Service in diesem Fall eine erfolgreiche Antwort gibt.

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                 &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Namensservice initialisieren

Diese Funktion wird von einer Namensservicekomponente bereitgestellt und vom Warteschlangenmanager beim Konfigurieren der Komponente gestartet. Es wird erwartet, dass sie MQZEP aufruft, um dem Warteschlangenmanager Informationen bereitzustellen.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_INIT_NAME.

Syntax

MQZ_INIT_NAME(*Hconfig*, *Options*, *QMgrName*, *ComponentDataLength*, *ComponentData*, *Version*, *CompCode*, *Reason*)

Parameter

Hconfig
Typ: MQHCONFIG - Eingabe

Konfigurationskennung. Diese Kennung stellt die spezielle Komponente dar, die gerade initialisiert wird. Sie muss von der Komponente beim Aufruf des Warteschlangenmanagers mit der MQZEP-Funktion verwendet werden.

Optionen
Typ: MQLONG - Eingabe

Initialisierungsoptionen. Folgende Werte sind zulässig:

MQZIO_PRIMARY
Primärinitialisierung.

MQZIO_SECONDARY
Sekundärinitialisierung.

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ComponentDataLength
Typ: MQLONG - Eingabe

Länge der Komponentendaten. Länge des Bereichs *ComponentData* in Byte. Diese Länge wird in den Daten der Komponentenkonfiguration definiert.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Dieser Parameter wird ausschließlich mit Nullen initialisiert, bevor die primäre Initialisierungsfunktion der Komponente aufgerufen wird. Der Warteschlangenmanager speichert diese Daten für diese spezielle Komponente. Alle Änderungen, die die von der Komponente bereitgestellten Funktionen (einschließlich der Initialisierungsfunktion) daran vornehmen, werden aufgezeichnet und beim nächsten Aufruf einer dieser Komponentenfunktionen angezeigt.

Die Länge dieses Datenbereichs wird vom Warteschlangenmanager im Parameter *ComponentDataLength* des Aufrufs MQZ_INIT_AUTHORITY übermittelt.

Version
Typ: MQLONG - Eingabe/Ausgabe

Versionsnummer. Bei der Eingabe für die Initialisierungsfunktion gibt dieser Wert die höchste Versionsnummer an, die der Warteschlangenmanager unterstützt. Die Initialisierungsfunktion muss diesen Wert gegebenenfalls in die Nummer der von ihr unterstützten Schnittstellenversion ändern. Wenn der Warteschlangenmanager bei der Rückgabe die von der Komponente zurückgegebene Version nicht unterstützt, ruft er die Funkton MQZ_TERM_NAME der Komponente auf verwendet diese Komponente nicht mehr.

Folgende Werte werden unterstützt:

MQZAS_VERSION_1
Version 1.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Initialisierung aus nicht definiertem Grund fehlgeschlagen.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Name einfügen

Diese Funktion wird von einer Namensservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um einen Eintrag in die angegebene Warteschlange einzufügen, der den Namen des Warteschlangenmanagers enthält, welcher der Eigner der Warteschlange ist. Wenn die Warteschlange bereits im Service definiert ist, schlägt der Aufruf fehl.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_INSERT_NAME.

Syntax

MQZ_INSERT_NAME(*QMgrName*, *QName*, *ResolvedQMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName

Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

Warteschlangenname

Typ: MQCHAR48 - Eingabe

Der Name der Warteschlange. Der Name der Warteschlange, in die ein Eintrag eingefügt werden soll. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

ResolvedQMgrName

Typ: MQCHAR48 - Eingabe

Aufgelöster Warteschlangenmanagername. Gibt den Namen des Warteschlangenmanagers an, in den die Warteschlange aufgelöst wird. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

ComponentData

Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Der Warteschlangenmanager speichert diese Daten für diese spezielle Komponente. Alle Änderungen, die die von der Komponente bereitgestellten Funktionen (einschließlich der

Initialisierungsfunktion) daran vornehmen, werden aufgezeichnet und beim nächsten Aufruf einer dieser Komponentenfunktionen angezeigt.

Der Warteschlangenmanager übergibt die Länge dieses Datenbereichs an den Parameter *ComponentDataLength* des Aufrufs von MQZ_INIT_NAME.

Continuation
Typ: MQLONG - Eingabe/Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Bei MQZ_INSERT_NAME versucht der Warteschlangenmanager nicht, eine andere Komponente zu starten, unabhängig vom Rückgabewert des Parameters *Continuation*.

Folgende Werte werden unterstützt:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Reason
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_Q_ALREADY_EXISTS
(2290, X'8F2') Warteschlangenobjekt bereits vorhanden.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie im Abschnitt API-Ursachencode.

C-Aufruf

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Name suchen

Diese Funktion wird von einer Namensservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, um den Namen des Warteschlangenmanagers abzurufen, der der Eigner einer bestimmten Warteschlange ist.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_LOOKUP_NAME.

Syntax

MQZ_LOOKUP_NAME(*QMgrName*, *QName*, *ResolvedQMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parameter

QMgrName
Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

Warteschlangenname
Typ: MQCHAR48 - Eingabe

Der Name der Warteschlange. Der Name der Warteschlange, für die ein Eintrag aufgelöst werden soll. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

ResolvedQMgrName
Typ: MQCHAR48 - Ausgabe

Aufgelöster Warteschlangenmanagername. Wenn die Funktion erfolgreich ausgeführt wird, ist dies der Name des Warteschlangenmanagers, der der Eigner der Warteschlange ist.

Der von der Servicekomponente zurückgegebene Name muss rechts mit Leerzeichen auf die vollständige Länge des Parameters aufgefüllt werden. Der Name darf nicht durch ein Nullzeichen beendet werden oder führende oder eingebettete Leerzeichen enthalten.

ComponentData
Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Der Warteschlangenmanager speichert diese Daten für diese spezielle Komponente. Alle Änderungen, die die von der Komponente bereitgestellten Funktionen (einschließlich der Initialisierungsfunktion) daran vornehmen, werden aufgezeichnet und beim nächsten Aufruf einer dieser Komponentenfunktionen angezeigt.

Der Warteschlangenmanager übergibt die Länge dieses Datenbereichs an den Parameter *ComponentDataLength* des Aufrufs von MQZ_INIT_NAME.

Continuation
Typ: MQLONG - Ausgabe

Von der Komponente festgelegter Fortsetzungsanzeiger. Bei MQZ_LOOKUP_NAME gibt der Warteschlangenmanager wie folgt an, ob eine andere Namensservicekomponente gestartet werden soll:

- Wenn *CompCode* den Wert MQCC_OK hat, werden keine weiteren Komponenten gestartet, unabhängig vom Rückgabewert des Parameters *Continuation*.

- Wenn *CompCode* nicht den Wert MQCC_OK hat, wird eine weitere Komponente gestartet, sofern der Wert für *Continuation* nicht MQZCI_STOP lautet.

Folgende Werte werden unterstützt:

MQZCI_DEFAULT
Die Fortsetzung hängt vom Warteschlangenmanager ab.

MQZCI_CONTINUE
Mit der nächsten Komponente fortfahren.

MQZCI_STOP
Nicht mit der nächsten Komponente fortfahren.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unerwarteter Fehler beim Zugriff auf den Service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

MQRC_UNKNOWN_Q_NAME
(2288, X'8F0') Name der Warteschlange wurde nicht gefunden.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Namensservice beenden

Diese Funktion wird von einer Namensservicekomponente bereitgestellt und vom Warteschlangenmanager gestartet, wenn er die Services dieser Komponente nicht mehr benötigt. Von dieser Funktion müssen alle für die Komponente erforderlichen Bereinigungsmaßnahmen durchgeführt werden.

Die Funktions-ID dieser Funktion (für MQZEP) ist MQZID_TERM_NAME.

Syntax

MQZ_TERM_NAME(*Hconfig*, *Options*, *QMgrName*, *ComponentData*, *CompCode*, *Reason*)

Parameter

Hconfig

Typ: MQHCONFIG - Eingabe

Konfigurationskennung. Diese Kennung stellt die Komponente dar, die beendet wird. Sie wird von der Komponente beim Aufrufen des Warteschlangenmanagers mit der Funktion MQZEP verwendet.

Optionen

Typ: MQLONG - Eingabe

Beendigungsoptionen. Folgende Werte sind zulässig:

MQZTO_PRIMARY
Primärbeendigung.

MQZTO_SECONDARY
Sekundäre Beendigung.

QMgrName

Typ: MQCHAR48 - Eingabe

Warteschlangenmanagername. Der Name des Warteschlangenmanagers, der die Komponente aufruft. Dieser Name wird bis zur vollständigen Parameterlänge rechts mit Leerzeichen aufgefüllt. Der Name wird nicht mit einem Nullzeichen beendet.

Der WS-Manager-Name wird zu Informationszwecken an die Komponente übermittelt. Die Berechtigungsserviceschnittstelle erfordert nicht, dass die Komponente den Namen auf eine bestimmte Weise verwendet.

ComponentData

Typ: MQBYTE×ComponentDataLength - Eingabe/Ausgabe

Komponentendaten. Der Warteschlangenmanager speichert diese Daten für diese spezielle Komponente. Alle Änderungen, die die von der Komponente bereitgestellten Funktionen (einschließlich der Initialisierungsfunktion) daran vornehmen, werden aufgezeichnet und beim nächsten Aufruf einer dieser Komponentenfunktionen angezeigt.

Die Komponentendaten befinden sich im gemeinsam genutzten Speicher und im Zugriff für alle Prozesse.

Der Warteschlangenmanager übergibt die Länge dieses Datenbereichs an den Parameter *ComponentDataLength* des Aufrufs von MQZ_INIT_NAME.

Wenn der Aufruf von MQZ_TERM_NAME abgeschlossen ist, verwirft der Warteschlangenmanager diese Daten.

CompCode

Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

Ursachencode, der *CompCode* qualifiziert.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_TERMINATION_FAILED
(2287, X'8FF') Beendigung aus nicht definiertem Grund fehlgeschlagen.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Der zugrunde liegende Service ist nicht verfügbar.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
               &Reason);
```

The parameters passed to the service are declared as follows:

```
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 - Anwendungskontext

Die MQZAC-Struktur wird im Aufruf von MQZ_AUTHENTICATE_USER für den Parameter *ApplicationContext* verwendet. Dieser Parameter gibt Daten zu der aufrufenden Anwendung an.

In *Tabelle 1* finden Sie eine Zusammenfassung der in der Struktur enthaltenen Felder.

Tabelle 596. Felder in MQZAC

Feld	**Beschreibung**
StrucId	Struktur-ID
Version	Strukturversionsnummer
ProcessId	Prozess-ID
ThreadId	Thread-ID
ApplName	Anwendungsname
UserID	Benutzer-ID
EffectiveUserID	Effektive Benutzer-ID
Environment	Umgebung
CallerType	Aufrufertyp
AuthenticationType	Authentifizierungstyp

Tabelle 596. Felder in MQZAC (Forts.)	
Feld	**Beschreibung**
BindType	Bindungstyp

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQZAC_STRUC_ID
ID für die Anwendungskontextstruktur.

Für die Programmiersprache C ist auch die Konstante MOZAC_STRUC_ID_ARRAY definiert; diese hat den gleichen Wert wie MOZAC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQZAC_VERSION_1
Anwendungskontextstruktur der Version 1. Die Konstante MQZAC_CURRENT_VERSION gibt die Versionsnummer der aktuellen Version an.

ProcessId
Typ: MQPID - Eingabe

Prozess-ID der Anwendung.

ThreadId
Typ: MQTID - Eingabe

Thread-ID der Anwendung.

ApplName
Typ: MQCHAR28 - Eingabe

Anwendungsname.

UserID
Typ: MQCHAR12 - Eingabe

Benutzer-ID. Auf UNIX-Systemen gibt dieses Feld die reale Benutzer-ID der Anwendung an. Unter Windows gibt dieses Feld die Benutzer-ID der Anwendung an.

EffectiveUserID
Typ: MQCHAR12 - Eingabe

Effektive Benutzer-ID. Auf UNIX-Systemen gibt dieses Feld die aktuelle Benutzer-ID der Anwendung an. Unter Windows ist dieses Feld leer.

Environment
Typ: MQLONG - Eingabe

Die Umgebung. Dieses Feld gibt die Umgebung an, aus der der Aufruf ausgeführt wurde. In diesem Feld sind folgende Werte möglich:

MQXE_COMMAND_SERVER
Befehlsserver

MQXE_MQSC
Befehlsinterpreter von **runmqsc**

MQXE_MCA
Nachrichtenkanalagent MQXE_OTHER

MQXE_OTHER
Nicht definierte Umgebung

CallerType
Typ: MQLONG - Eingabe

Aufrufertyp. Dieses Feld gibt den Typ des Programms an, das den Aufruf ausgeführt hat. In diesem Feld sind folgende Werte möglich:

MQXACT_EXTERNAL
Der Aufruf wurde außerhalb des Warteschlangenmanagers ausgeführt.

MQXACT_INTERNAL
Der Aufruf wurde innerhalb des Warteschlangenmanagers ausgeführt.

AuthenticationType
Typ: MQLONG - Eingabe

Authentifizierungstyp. Dieses Feld gibt den Authentifizierungstyp an, der gerade ausgeführt wird. In diesem Feld sind folgende Werte möglich:

MQZAT_INITIAL_CONTEXT
Der Authentifizierungsaufruf beruht auf dem initialisierten Benutzerkontext. Dieser Wert wird während eines Aufrufs von MQCONN oder MQCONNX verwendet.

MQZAT_CHANGE_CONTEXT
Der Authentifizierungsaufruf beruht auf dem geänderten Benutzerkontext. Dieser Wert wird verwendet, wenn der Nachrichtenkanalagent (MCA) den Benutzerkontext ändert. Übergeordnetes Thema: MQZAC -

BindType
Typ: MQLONG - Eingabe

Bindungstyp. Dieses Feld gibt den verwendeten Bindungstyp an. In diesem Feld sind folgende Werte möglich:

MQCNO_FASTPATH_BINDING
Fastpath-Bindung.

MQCNO_SHARED_BINDING
Gemeinsam genutzte Bindung.

MQCNO_ISOLATED_BINDING
Isolierte Bindung.

Deklaration in Programmiersprache C
Deklarieren Sie die Felder der Struktur wie folgt:

```
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 - Berechtigungsdaten

Die MQZAD-Struktur wird im MQZ_ENUMERATE_AUTHORITY_DATA-Aufruf für zwei Parameter verwendet, eine Eingabe und eine Ausgabe.

- MQZAD wird für den Parameter *Filter* verwendet, der als Eingabe für den Aufruf dient. Dieser Parameter gibt die Auswahlkriterien an, die bei der Auswahl der vom Aufruf zurückgegebenen Berechtigungsdaten verwendet werden sollen.
- Darüber hinaus wird MQZAD für den Parameter *AuthorityBuffer* verwendet, der eine Ausgabe des Aufrufs darstellt. Dieser Parameter gibt die Berechtigungen für eine Kombination aus Profilname, Objekttyp und Entität an.

In *Tabelle 1* finden Sie eine Zusammenfassung der in der Struktur enthaltenen Felder.

Tabelle 597. Felder in MQZAD

Feld	Beschreibung
StrucId	Struktur-ID
Version	Strukturversionsnummer
ProfileName	Prozess-ID
ObjectType	Thread-ID
Berechtigung	Anwendungsname
EntityDataPtr	Benutzer-ID
EntityType	Umgebung
Options	Aufrufertyp

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQZAC_STRUC_ID
ID für die Anwendungskontextstruktur.

Für die Programmiersprache C ist auch die Konstante MOZAC_STRUC_ID_ARRAY definiert; diese hat den gleichen Wert wie MOZAC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQZAC_VERSION_1
Anwendungskontextstruktur der Version 1. Die Konstante MQZAC_CURRENT_VERSION gibt die Versionsnummer der aktuellen Version an.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQZAD_CURRENT_VERSION
Aktuelle Version der Berechtigungsdatenstruktur.

ProfileName
Typ: MQCHAR48 - Eingabe

Profilname.

Für den Parameter *Filter* ist dieses Feld der Profilname, für den Berechtigungsdaten erforderlich sind. Wenn der Name bis zum Feldende oder zum ersten Nullzeichen ganz leer ist, werden Berechtigungsdaten für alle Profilnamen zurückgegeben.

Für den Parameter *AuthorityBuffer* ist dieses Feld der Name eines Profils, das die angegebenen Auswahlkriterien erfüllt.

ObjectType
Typ: MQLONG - Eingabe

Objekttyp.

Für den Parameter *Filter* ist dieses Feld der Objekttyp, für den Berechtigungsdaten erforderlich sind. Wenn der Wert MQOT_ALL lautet, werden Berechtigungsdaten für alle Objekttypen zurückgegeben.

Für den Parameter *AuthorityBuffer* ist dieses Feld der Objekttyp, auf den das mit dem Parameter *ProfileName* angegebene Profil angewendet wird.

Einer der folgende Werte ist möglich (für den Parameter *Filter* zusätzlich noch der Wert MQOT_ALL):

MQOT_AUTH_INFO
Authentifizierungsdaten

MQOT_CHANNEL
Kanal

MQOT_CLNTCONN_CHANNEL
Clientverbindungskanal

MQOT_LISTENER
Empfangsprogramm

MQOT_NAMELIST
Namensliste

MQOT_PROCESS
Prozessdefinition

MQOT_Q
Warteschlange

MQOT_Q_MGR
Warteschlangenmanager

MQOT_SERVICE
Service

Berechtigung
Typ: MQLONG - Eingabe

Berechtigung.

Für den Parameter *Filter* wird dieses Feld ignoriert.

Für den Parameter *AuthorityBuffer* stellt dieses Feld die Berechtigungen dar, die die Entität für die mit *ProfileName* und *ObjectType* angegebenen Objekte besitzt. Wenn die Entität nur über eine Berechtigung verfügt, entspricht das Feld dem jeweiligen Berechtigungswert (MQZAO_*-Konstante). Bei mehreren Berechtigungen ist das Feld das bitweise ODER der entsprechenden MQZAO_*-Konstanten.

EntityDataPtr
Typ: PMQZED - Eingabe

Adresse der MQZED-Struktur, über die eine Entität identifiziert wird.

Für den Parameter *Filter* verweist dieses Feld auf eine MQZED-Struktur, die die Entität angibt, für die Berechtigungsdaten erforderlich sind. Wenn *EntityDataPtr* der Nullzeiger ist, werden Berechtigungsdaten für alle Entitäten zurückgegeben.

Für den Parameter *AuthorityBuffer* verweist dieses Feld auf eine MQZED-Struktur, die die Entität angibt, für die Berechtigungsdaten zurückgegeben wurden.

EntityType
Typ: MQLONG - Eingabe

Entitätstyp.

Für den Parameter *Filter* gibt dieses Feld den Entitätstyp an, für den Berechtigungsdaten erforderlich sind. Bei Angabe von MQZAET_NONE werden die Berechtigungsdaten für alle Entitätstypen zurückgegeben.

Für den Parameter *AuthorityBuffer* gibt dieses Feld den Entitätstyp an, der von der MQZED-Struktur angegeben wird, auf die der Parameter *EntityDataPtr* verweist.

Einer der folgende Werte ist möglich (für den Parameter *Filter* zusätzlich noch der Wert MQZAET_NONE):

MQZAET_PRINCIPAL
Principal

MQZAET_GROUP
Gruppe

Options
Typ: MQAUTHOPT - Eingabe

Optionen. Dieses Feld gibt die Optionen an, mit denen sich die angezeigten Profile steuern lassen. Es muss einer der folgenden Werte angegeben werden:

MQAUTHOPT_NAME_ALL_MATCHING
Zeigt alle Profile an.

MQAUTHOPT_NAME_EXPLICIT
Zeigt die Profile an, deren Namen mit dem im Feld *ProfileName* angegebenen Namen übereinstimmen.

Außerdem muss auch eine der folgenden Optionen angegeben werden:

MQAUTHOPT_ENTITY_SET
Zeigt alle Profile an, mit deren Hilfe die kumulative Berechtigung der Entität für das vom Parameter *ProfileName* angegebene Objekt berechnet wird. Der Parameter *ProfileName* darf keine Platzhalterzeichen enthalten.

Wenn die angegebene Entität ein Principal ist, wird für die einzelnen Mitglieder der Gruppe {Entität, Gruppen} das Profil angezeigt, das sich am besten auf das Objekt anwenden lässt.

Wenn die angegebene Entität eine Gruppe ist, wird das am besten passende Profil aus der Gruppe angezeigt, die für das Objekt gilt.

Wird dieser Wert angegeben, dürfen die Werte von *ProfileName*, *ObjectType*, *EntityType* sowie der Entitätsname, der in der MQZED-Struktur *EntityDataPtr* angegeben wird, nicht nur aus Leerzeichen bestehen.

Wenn Sie MQAUTHOPT_NAME_ALL_MATCHING angegeben haben, können Sie auch den folgenden Wert angeben:

MQAUTHOPT_ENTITY_EXPLICIT
Zeigt die Profile an, deren Entitätsnamen mit dem in der MQZED-Struktur *EntityDataPtr* angegebenen Entitätsnamen übereinstimmen.

Deklaration in Programmiersprache 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 */
};
```

Felder

MQZED - Entitätsdeskriptor

Mit der MQZED-Struktur wird in verschiedenen Berechtigungsserviceaufrufen die Entität angegeben, für die Berechtigungen geprüft werden sollen.

In *Tabelle 1* finden Sie eine Zusammenfassung der in der Struktur enthaltenen Felder.

Tabelle 598. Felder in MQZED

Feld	Beschreibung
StrucId	Struktur-ID
Version	Version
EntityName Ptr	Entitätsname
EntityDomainPtr	Verweis auf die Entitätsdomäne
SecurityId	Sicherheits-ID
CorrelationPtr	Korrelationsverweis

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQZED_STRUC_ID
ID für die Entitätsdeskriptorstruktur.

Für die Programmiersprache C ist auch die Konstante MQZED_STRUC_ID_ARRAY definiert; diese hat den gleichen Wert wie MQZED_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQZED_VERSION_1
Entitätsdeskriptorstruktur der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQZED_CURRENT_VERSION
Aktuelle Version der Entitätsdeskriptorstruktur.

EntityNamePtr
Typ: PMQCHAR - Eingabe

Profilname.

Adresse des Entitätsnamens. Dies ist ein Verweis auf den Namen der Entität, deren Berechtigung geprüft werden soll.

EntityDomainPtr
Typ: PMQCHAR - Eingabe

Adresse des Entitätsdomänennamens. Dies ist ein Verweis auf den Namen der Domäne, in der die Definition der Entität enthalten ist, deren Berechtigung geprüft werden soll.

SecurityId
Typ: MQBYTE40 - Eingabe

Berechtigung.

Sicherheits-ID. Dies ist die Sicherheits-ID, deren Berechtigung geprüft werden soll.

CorrelationPtr
Typ: MQPTR - Eingabe

Korrelationsverweis. Vereinfacht die Übergabe von Korrelationsdaten zwischen der Funktion zur Benutzerauthentifizierung und anderen einschlägigen OAM-Funktionen.

Deklaration in Programmiersprache 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 */
```

Felder

MQZEP - Eingangspunkt der Komponente hinzufügen

Diese Funktion wird während der Initialisierung von einer Servicekomponente gestartet, um einen Eingangspunkt zum Eingangspunktvektor für diese Servicekomponente hinzuzufügen.

Syntax
MQZEP (*Hconfig*, *Funktion*, *EntryPoint*, *CompCode*, *Reason*)

Parameter

Hconfig
Typ: MQHCONFIG - Eingabe

Konfigurationskennung. Diese Kennung stellt die Komponente dar, die für diesen installierbaren Service konfiguriert wird. Sie muss mit der Komponente identisch sein, die beim Komponenteninitialisierungsaufruf vom Warteschlangenmanager an die Komponentenkonfigurationsfunktion übergeben wird.

FUNCTION
Typ: MQLONG - Eingabe

Funktions-ID. Die gültigen Werte für diesen Parameter sind für jeden installierbaren Service definiert.

Wird MQZEP mehrmals für eine Funktion aufgerufen, stellt der letzte Aufruf den zu verwendenden Eingangspunkt bereit.

EntryPoint
Typ: PMQFUNC - Eingabe

Funktionseingangspunkt. Dies ist die Adresse des Eingangspunkts, der von der Komponente für die Ausführung der Funktion bereitgestellt wird.

Der Wert NULL ist gültig und zeigt an, dass die Funktion nicht von dieser Komponente bereitgestellt wird. NULL wird für Eingangspunkte angenommen, die nicht mit MQZEP definiert werden.

CompCode
Typ: MQLONG - Ausgabe

Beendigungscode. Folgende Werte sind zulässig:

MQCC_OK
Erfolgreiche Fertigstellung.

MQCC_FAILED
Aufruf fehlgeschlagen.

Ursache
Typ: MQLONG - Ausgabe

CompCode mit Ursachencode.

Wenn *CompCode* den Wert MQCC_OK aufweist:

MQRC_NONE
(0, X'000') Keine Ursache zurückzumelden

Wenn *CompCode* auf MQCC_FAILED gesetzt ist:

MQRC_FUNCTION_ERROR
(2281, X'8E9') Funktions-ID ungültig.

MQRC_HCONFIG_ERROR
(2280, X'8E8') Konfigurationskennung ungültig.

Weitere Informationen zu diesen Ursachencodes finden Sie unter API-Ursachencodes.

C-Aufruf

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Deklarieren Sie die Parameter wie folgt:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQZFP - Parameter freigeben

Die MQZFP-Struktur wird im MQZ_FREE_USER-Aufruf für den Parameter *FreeParms* verwendet. Dieser Parameter gibt Daten für die freizugebende Ressource an.

In *Tabelle 1* finden Sie eine Zusammenfassung der in der Struktur enthaltenen Felder.

Tabelle 599. Felder in MQZFP

Feld	Beschreibung
StrucId	Struktur-ID
Version	Version
Reserved	Reserviertes Feld
CorrelationPtr	Korrelationsverweis

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQZIC_STRUC_ID
ID für Identitätskontextstruktur. Für die Programmiersprache C ist auch die Konstante MOZIC_STRUC_ID_ARRAY definiert; diese hat den gleichen Wert wie MOZIC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQZFP_VERSION_1
Parameterfreigabestruktur der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQZFP_CURRENT_VERSION
Aktuelle Version der freien Parameterstruktur.

Reserved
Typ: MQBYTE8 - Eingabe

Reserviertes Feld. Der Anfangswert ist null.

CorrelationPtr
Typ: MQPTR - Eingabe

Korrelationsverweis. Die Adresse der Korrelationsdaten zu der freizugebenden Ressource.

Deklaration in Programmiersprache 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 */
};
```

Felder

MQZIC - Identitätskontext

Die MQZIC-Struktur wird im Aufruf von MQZ_AUTHENTICATE_USER für den Parameter *IdentityContext* verwendet.

Die MQZIC-Struktur enthält Identitätskontextdaten zur Identifizierung des Benutzers einer Anwendung, der als erster die Nachricht in eine Warteschlange einreiht:

- Der Warteschlangenmanager trägt im Feld *UserIdentifier* einen Namen zur Angabe des Benutzers ein. Wie der Warteschlangenmanager dabei vorgehen kann, richtet sich nach der Umgebung, in der die Anwendung ausgeführt wird.
- Der Warteschlangenmanager trägt im Feld *AccountingToken* ein Token oder eine Zahl ein, das bzw. die er aus der Anwendung ermittelt hat, welche die Nachricht eingereiht hat.
- Anwendungen können das Feld *ApplIdentityData* für Zusatzinformationen verwenden, die sie zu dem Benutzer eingeben wollen (beispielsweise ein verschlüsseltes Kennwort).

Entsprechend berechtigte Anwendungen können den Identitätskontext mithilfe der Funktion MQZ_AUTHENTICATE_USER festlegen.

Eine Windows -Systemsicherheits-ID (SID) wird im Feld *AccountingToken* gespeichert, wenn eine Nachricht unter WebSphere MQ für Windowserstellt wird. Mit der SID können Sie das Feld *UserIdentifier* ergänzen und die Berechtigungsnachweise eines Benutzers erstellen.

In *Tabelle 1* finden Sie eine Zusammenfassung der in der Struktur enthaltenen Felder.

Tabelle 600. Felder in MQZIC	
Feld	**Beschreibung**
StrucId	Struktur-ID
Version	Version
UserIdentifier	Benutzer-ID
AccountingToken	Abrechnung
ApplIdentityData	Anwendungsidentitätsdaten

Felder

StrucId
Typ: MQCHAR4 - Eingabe

Struktur-ID. Der Wert lautet wie folgt:

MQZIC_STRUC_ID
ID für Identitätskontextstruktur. Für die Programmiersprache C ist auch die Konstante MOZIC_STRUC_ID_ARRAY definiert; diese hat den gleichen Wert wie MOZIC_STRUC_ID, aber es handelt sich dabei um eine Gruppe von Zeichen, nicht um eine Zeichenfolge.

Version
Typ: MQLONG - Eingabe

Strukturversionsnummer. Der Wert lautet wie folgt:

MQZIC_VERSION_1
Identitätskontextstruktur der Version 1.

Die folgende Konstante definiert die Nummer der aktuellen Version:

MQZIC_CURRENT_VERSION
Aktuelle Version der Identitätskontextstruktur.

UserIdentifier
Typ: MQCHAR12 - Eingabe

Benutzer-ID. Diese Information ist Teil des Identitätskontexts einer Nachricht. *UserIdentifier* gibt die Benutzer-ID der Anwendung an, die die Nachricht generiert hat. Der Warteschlangenmanager behandelt diese Informationen als Zeichendaten, ohne jedoch ihr Format zu definieren. Weitere Informationen zum Feld *UserIdentifier* finden Sie unter „UserIdentifier (MQCHAR12)" auf Seite 447.

AccountingToken
Typ: MQBYTE32 - Eingabe

Berechnungs-Token. Diese Information ist Teil des Identitätskontexts einer Nachricht. Mit der Angabe von *AccountingToken* kann eine Anwendung eine Aufgabe infolge der entsprechend zu ladenden Nachricht als erledigt betrachten. Der Warteschlangenmanager behandelt diese Informationen als Bitzeichenfolge, ohne jedoch ihren Inhalt zu prüfen. Weitere Informationen zum Feld *AccountingToken* finden Sie unter „AccountingToken (MQBYTE32)" auf Seite 402.

ApplIdentityData
Typ: MQCHAR32 - Eingabe

Identitätsbezogene Anwendungsdaten. Diese Information ist Teil des Identitätskontexts einer Nachricht. ApplIdentityData ist eine von der Anwendungssuite definierte Information, die zur Bereitstellung zusätzlicher Informationen über den Ursprung der Nachricht verwendet werden kann. Es kann beispielsweise von Anwendungen eingestellt werden, die mit geeigneten Benutzerberechtigungen ausgeführt werden, um anzuzeigen, ob die Identitätsdaten vertrauenswürdig sind. Weitere Informationen zum Feld "ApplIdentityData" finden Sie unter „ApplIdentityData (MQCHAR32)" auf Seite 404.

Deklaration in Programmiersprache 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 */
};
```

Felder

Referenzmaterial für IBM WebSphere MQ Bridge for HTTP

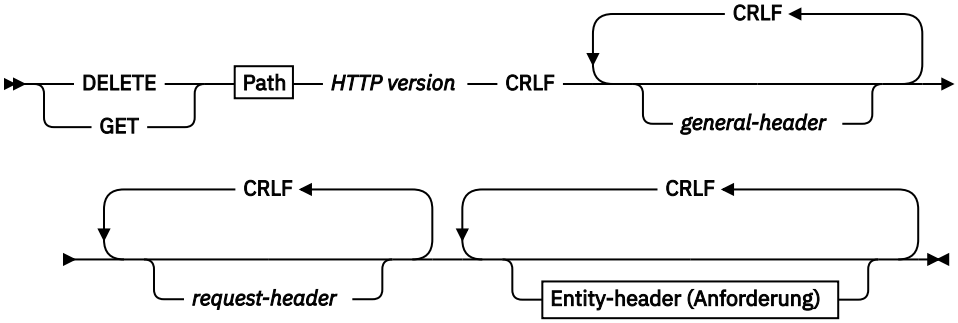
Referenzthemen für IBM WebSphere MQ Bridge for HTTP in alphabetischer Reihenfolge

HTTP DELETE: WebSphere MQ Bridge for HTTP-Befehl

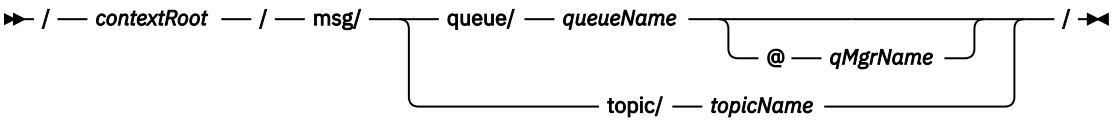
Bei einer HTTP-Operation vom Typ **DELETE** wird eine Nachricht aus einer WebSphere MQ-Warteschlange oder eine Veröffentlichung aus einem Thema abgerufen. Dabei wird die Nachricht aus der Warteschlange entfernt. Wenn die Veröffentlichung beibehalten wird, wird sie nicht entfernt. Eine Antwortnachricht mit Informationen zur Nachricht wird an den Client zurückgesendet.

Syntax

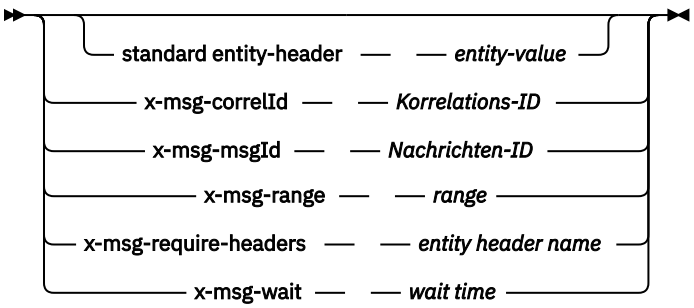
Anforderung



Path



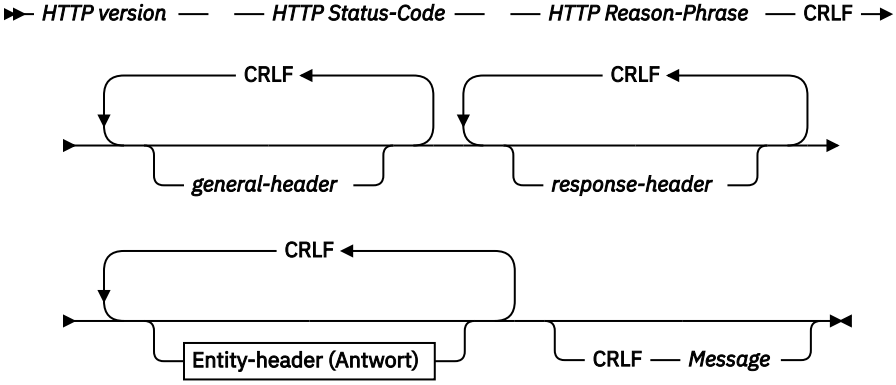
entity-header (Request)



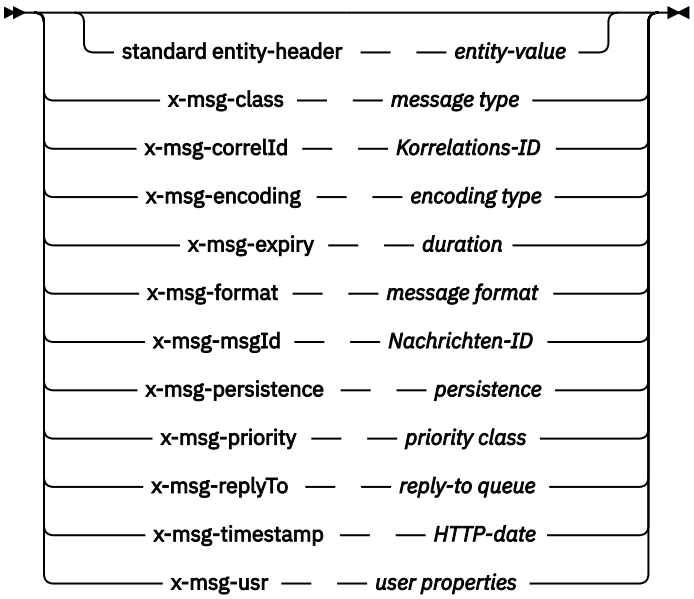
Anmerkung:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Antwort



entity-header (Antwort)



Anforderungsparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

request-header
Informationen finden Sie unter HTTP/1.1 - 5.3 Request Header Fields ("request-header"-Felder). Das Host-Feld ist in einer HTTP/1.1-Anforderung obligatorisch. Es wird häufig von dem Tool, das Sie zum Erstellen einer Clientanforderung verwenden, automatisch eingefügt.

entity-header (Request)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Anforderungen aufgeführten Entitätsheader.

Antwortparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

response-header
Informationen finden Sie unter HTTP/1.1 - 6.2 Response Header Fields ("response-header"-Felder).

entity-header (Antwort)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Antworten aufgeführten Entitäts- oder Antwortheader. `Content-Length` ist in einer Antwort immer vorhanden. Er ist auf null gesetzt, wenn kein Nachrichtentext vorhanden ist.

Nachricht
Nachrichtentext.

Beschreibung

Wenn die HTTP-Anforderung vom Typ **DELETE** erfolgreich ist, enthält die Antwortnachricht die aus der WebSphere MQ-Warteschlange abgerufenen Daten. Die Anzahl der im Nachrichtentext enthaltenen Bytes wird im HTTP `Content-Length`-Header zurückgegeben. Der Statuscode für die HTTP-Antwort ist auf `200 OK` festgelegt. Wenn `x-msg-range` mit `0` oder `0-0` angegeben wird, lautet der Statuscode der HTTP-Antwort `204 No Content` (kein Inhalt).

Wenn die HTTP-Anforderung vom Typ **DELETE** nicht erfolgreich ist, enthält die Antwort eine WebSphere MQ Bridge for HTTP-Fehlermeldung und einen HTTP-Statuscode.

Beispiel für HTTP DELETE

Mit HTTP **DELETE** wird eine Nachricht aus einer Warteschlange oder eine Veröffentlichung abgerufen und gelöscht. Das **HTTPDELETE** Java-Beispiel ist ein Beispiel für eine HTTP **DELETE** -Anforderung, die eine Nachricht aus einer Warteschlange liest. Anstatt Java zu verwenden, können Sie die HTTP-Anforderung **DELETE** in einem Browserformular oder mithilfe eines AJAX-Toolkits erstellen.

In Abbildung 37 auf Seite 1244 ist eine HTTP-Anforderung zum Löschen der nächsten Nachricht in der Warteschlange `myQueue` dargestellt. Als Antwort wird der Nachrichtentext an den Client zurückgegeben. In WebSphere MQ ist HTTP **DELETE** ein Abruf mit Löschen.

Die Anforderung enthält den HTTP-Anforderungsheader `x-msg-wait`, der WebSphere MQ Bridge for HTTP mitteilt, wie lange auf die Ankunft einer Nachricht in der Warteschlange gewartet werden soll. Die Anforderung enthält zudem den Anforderungsheader `x-msg-require-headers`, der angibt, dass der Client in der Antwort die Korrelations-ID der Nachricht erhalten soll.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Abbildung 37. Beispiel für eine HTTP-Anforderung vom Typ **DELETE***

Abbildung 38 auf Seite 1245 ist die Antwort, die an den Client zurückgegeben wird. Die Korrelations-ID wird an den Client zurückgegeben, wie in `x-msg-require-headers` der Anforderung angefordert.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

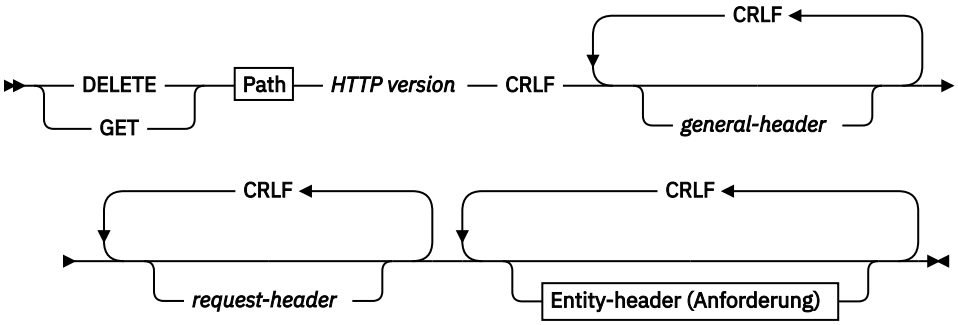
*Abbildung 38. Beispiel für eine HTTP-Antwort vom Typ **DELETE***

HTTP GET: WebSphere MQ Bridge for HTTP-Befehl

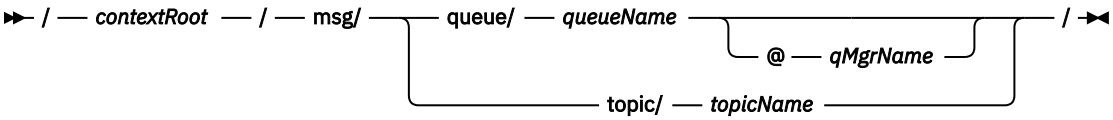
Bei einer HTTP-Operation vom Typ **GET** wird eine Nachricht aus einer WebSphere MQ-Warteschlange abgerufen. Die Nachricht bleibt in der Warteschlange. Die HTTP-Operation vom Typ **GET** entspricht dem Durchsuchen einer WebSphere MQ-Warteschlange.

Syntax

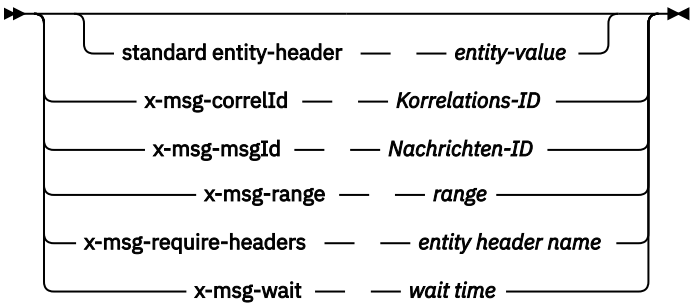
Anforderung



Path



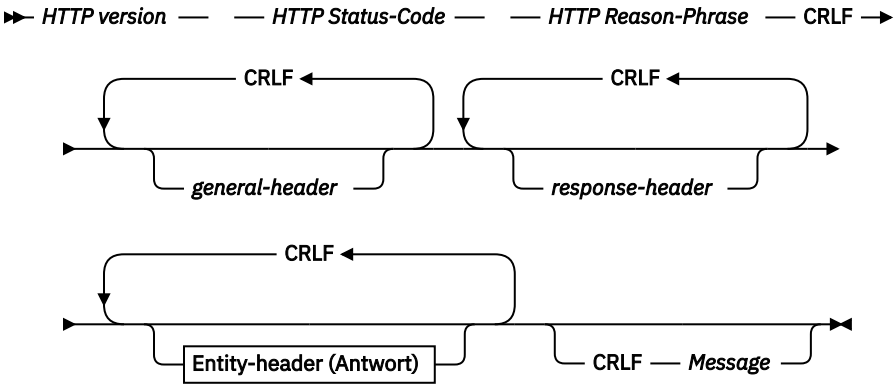
entity-header (Request)



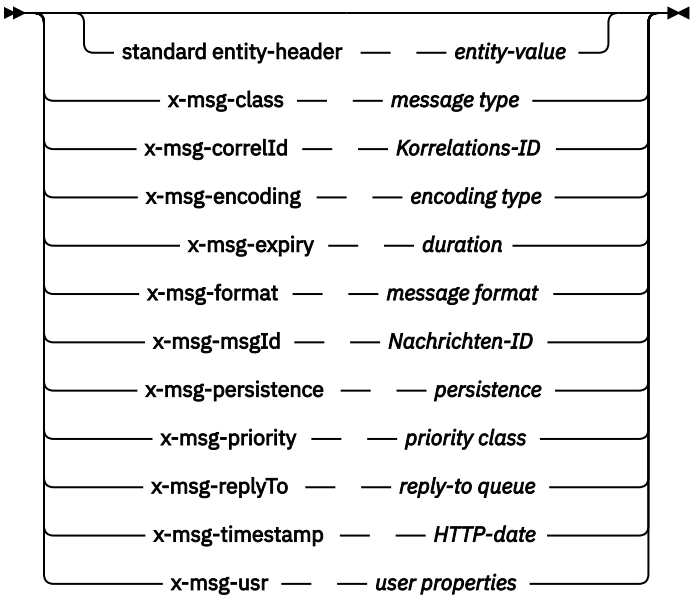
Anmerkung:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Antwort



entity-header (Antwort)



Anforderungsparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

request-header
Informationen finden Sie unter HTTP/1.1 - 5.3 Request Header Fields ("request-header"-Felder). Das Host-Feld ist in einer HTTP/1.1-Anforderung obligatorisch. Es wird häufig von dem Tool, das Sie zum Erstellen einer Clientanforderung verwenden, automatisch eingefügt.

entity-header (Request)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Anforderungen aufgeführten Entitätsheader.

Antwortparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

response-header
Informationen finden Sie unter HTTP/1.1 - 6.2 Response Header Fields ("response-header"-Felder).

entity-header (Antwort)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Antworten aufgeführten Entitäts- oder Antwortheader. `Content-Length` ist in einer Antwort immer vorhanden. Er ist auf null gesetzt, wenn kein Nachrichtentext vorhanden ist.

Nachricht
Nachrichtentext.

Beschreibung

Wenn die HTTP-Anforderung vom Typ **GET** erfolgreich ist, enthält die Antwortnachricht die aus der WebSphere MQ-Warteschlange abgerufenen Daten. Die Anzahl der im Nachrichtentext enthaltenen Bytes wird im HTTP `Content-Length`-Header zurückgegeben. Der Statuscode für die HTTP-Antwort ist auf `200 OK` festgelegt. Wenn `x-msg-range` mit `0` oder `0-0` angegeben wird, lautet der Statuscode der HTTP-Antwort `204 No Content` (kein Inhalt).

Wenn die HTTP-Anforderung vom Typ **GET** nicht erfolgreich ist, enthält die Antwort eine WebSphere MQ Bridge for HTTP-Fehlermeldung und einen HTTP-Statuscode.

Beispiel für HTTP GET

Mit der HTTP-Anforderung vom Typ **GET** wird eine Nachricht aus einer Warteschlange abgerufen. Die Nachricht bleibt in der Warteschlange. Aus Sicht von WebSphere MQ ist HTTP **GET** eine Anforderung zum Durchsuchen. Sie können eine HTTP-Anforderung vom Typ **GET** mithilfe eines Java-Clients, eines Browserformulars oder eines AJAX-Toolkits erstellen.

In Abbildung 39 auf Seite 1247 ist eine HTTP-Anforderung zum Durchsuchen der nächsten Nachricht in der Warteschlange `myQueue` dargestellt.

Die Anforderung enthält den HTTP-Anforderungsheader `x-msg-wait`, der WebSphere MQ Bridge for HTTP mitteilt, wie lange auf die Ankunft einer Nachricht in der Warteschlange gewartet werden soll. Die Anforderung enthält zudem den Anforderungsheader `x-msg-require-headers`, der angibt, dass der Client in der Antwort die Korrelations-ID der Nachricht erhalten soll.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Abbildung 39. Beispiel für eine HTTP-Anforderung vom Typ **GET***

Abbildung 40 auf Seite 1248 ist die Antwort, die an den Client zurückgegeben wird. Die Korrelations-ID wird an den Client zurückgegeben, wie in `x-msg-require-headers` der Anforderung angefordert.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

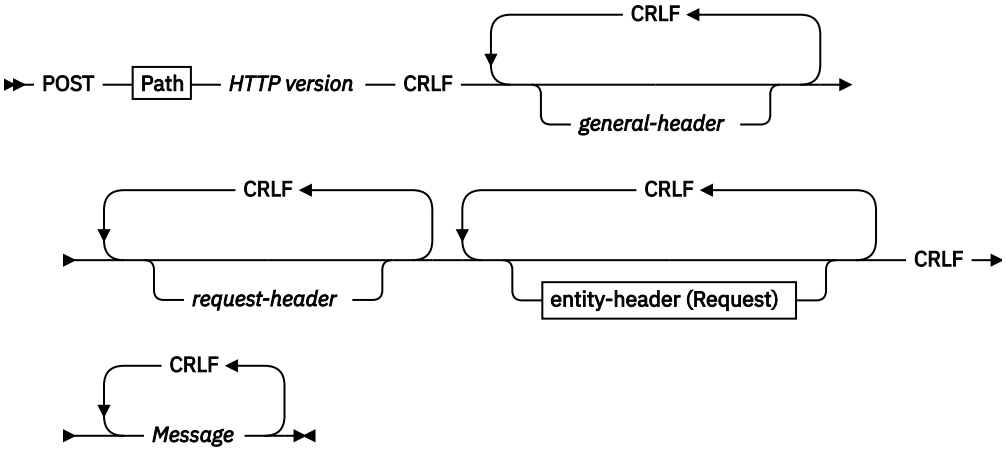
*Abbildung 40. Beispiel für eine HTTP-Antwort vom Typ **GET***

HTTP POST: WebSphere MQ Bridge for HTTP-Befehl

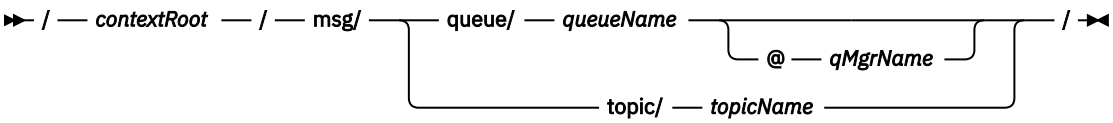
Bei einer HTTP-Operation vom Typ **POST** wird eine Nachricht in eine WebSphere MQ-Warteschlange gestellt oder in einem Thema veröffentlicht.

Syntax

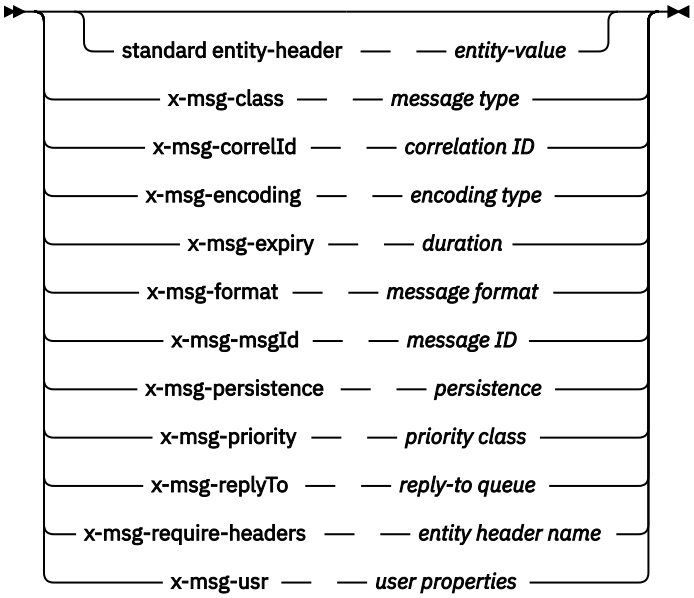
Request



Path



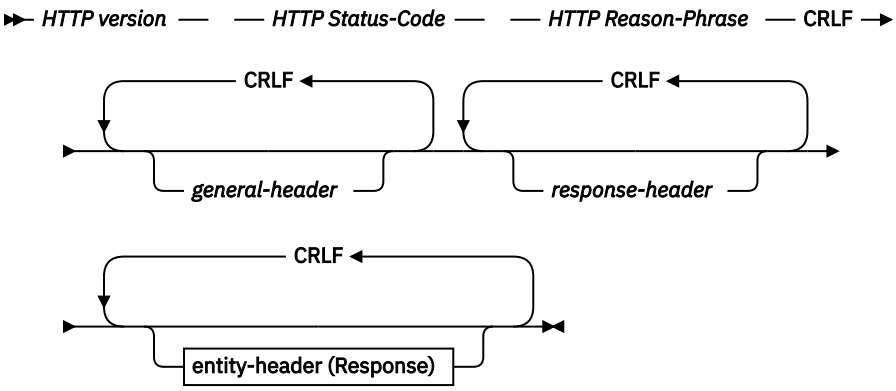
entity-header (Request)



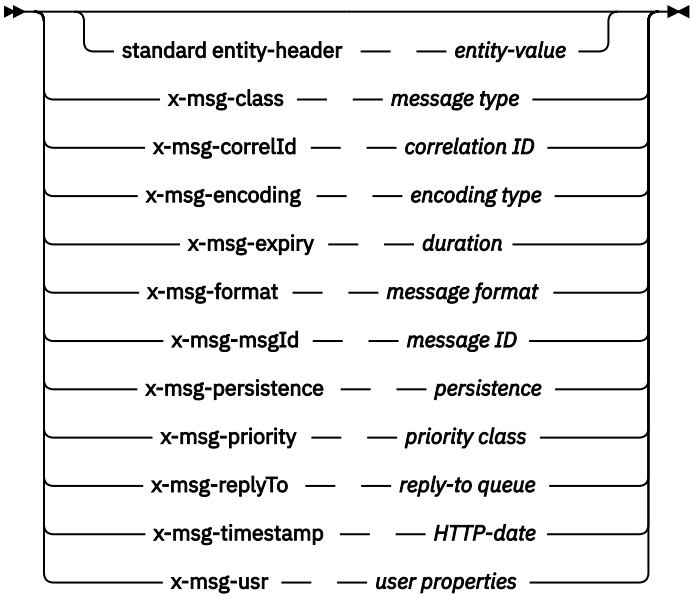
Anmerkung:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Response



entity-header (Response)



Anforderungsparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

request-header
Informationen finden Sie unter HTTP/1.1 - 5.3 Request Header Fields ("request-header"-Felder). Das Host-Feld ist in einer HTTP/1.1-Anforderung obligatorisch. Es wird häufig von dem Tool, das Sie zum Erstellen einer Clientanforderung verwenden, automatisch eingefügt.

entity-header (Request)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Anforderungen aufgeführten Entitätsheader. `Content-Length` und `Content-Type` sollten in eine Anforderung eingefügt werden und werden häufig automatisch von dem Tool eingefügt, das Sie zum Erstellen einer Clientanforderung verwenden. Wenn der `Content-Type`-Header angegeben wird, muss er mit dem im benutzerdefinierten Entitätsheader `x-msg-class` definierten Typ übereinstimmen.

Nachricht
Die Nachricht, die in die Warteschlange gestellt wird, bzw. die Veröffentlichung, die in einem Thema veröffentlicht wird.

Antwortparameter

PATH
Siehe „URI-Format” auf Seite 1278.

HTTP-Version
HTTP-Version; Beispiel: `HTTP/1.1`

general-header
Informationen finden Sie unter HTTP/1.1 - 4.5 General Header Fields ("general-header"-Felder).

response-header
Informationen finden Sie unter HTTP/1.1 - 6.2 Response Header Fields ("response-header"-Felder).

entity-header (Antwort)
Informationen finden Sie unter HTTP/1.1 - 7.1 Entity Header Fields ("entity-header"-Felder). Einer der im Syntaxdiagramm für Antworten aufgeführten Entitäts- oder Antwortheader. `Content-Length` ist in einer Antwort immer vorhanden. Er ist auf null gesetzt, wenn kein Nachrichtentext vorhanden ist.

Beschreibung

Wenn kein `x-msg-usr`-Header vorhanden ist und die Nachrichtenklasse BYTES oder TEXT lautet, weist die in die Warteschlange gestellte Nachricht keinen `MQRFH2`-Header auf.

Verwenden Sie HTTP-Entitäts- und HTTP-Anforderungsheader in der HTTP-Anforderung vom Typ **POST**, um die Eigenschaften der in die Warteschlange eingereihten Nachricht festzulegen. Sie können auch `x-msg-require-headers` verwenden, um abzufragen, welche Header in der Antwortnachricht zurückgegeben werden.

Wenn die HTTP-Anforderung vom Typ **POST** erfolgreich ist, ist die Entität der Antwortnachricht leer und der zugehörige Wert für "Content-Length" ist null. Der HTTP-Statuscode lautet `200 OK`.

Wenn die HTTP-Anforderung vom Typ **POST** nicht erfolgreich ist, enthält die Antwort eine WebSphere MQ Bridge for HTTP-Fehlermeldung und einen HTTP-Statuscode. Die WebSphere MQ-Nachricht wird nicht in die Warteschlange eingereiht oder im Thema veröffentlicht.

Beispiel für HTTP-POST

HTTP **POST** reiht eine Nachricht in eine Warteschlange oder eine Veröffentlichung zu einem Thema ein. Das Java-Beispiel **HTTPPOST** ist ein Beispiel für eine HTTP-**POST**-Anforderung einer Nachricht an eine Warteschlange. Anstelle von Java können Sie eine HTTP-**POST**-Anforderung mithilfe eines Browserformulars oder eines AJAX-Toolkits erstellen.

In Abbildung 41 auf Seite 1251 ist eine HTTP-Anforderung dargestellt, mit der eine Nachricht in die Warteschlange `myQueue` gestellt wird. Diese Anforderung enthält den HTTP-Header `x-msg-correlId` zum Festlegen der Korrelations-ID der WebSphere MQ-Nachricht.

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Abbildung 41. Beispiel für eine HTTP-**POST**-Anforderung an eine Warteschlange*

Abbildung 42 auf Seite 1251 zeigt die Antwort an, die dem Client zurückgegeben wird. Es ist kein Antwortinhalt vorhanden.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Abbildung 42. Beispiel für eine HTTP-Antwort vom Typ POST

HTTP-Header

WebSphere MQ Bridge for HTTP unterstützt benutzerdefinierte HTTP-Anforderungsheader und -Entitätsheader sowie eine Reihe von HTTP-Standardheadern.

Bei HTTP wird allen benutzerdefinierten Headern ein `x-`, den WebSphere MQ Bridge for HTTP-Header ein `x-msg-` vorangestellt. Beispiel: Verwenden Sie `x-msg-priority`, um den Prioritätsheader festzulegen.

Anmerkung:

- Bei den meisten Headerwerten muss die Groß-/Kleinschreibung beachtet werden. Beispiel: Bei der Verwendung des `msgId`-Headers ist `NONE` ein Schlüsselwort, während `none` eine `msgID` ist.
- Falsch geschriebene Header werden ignoriert.

Benutzerdefinierte HTTP-Entitätsheader

Die benutzerdefinierten HTTP-Entitätsheader enthalten Informationen zu WebSphere MQ-Nachrichten. Mithilfe von Entitätsheadern können Sie Werte im Nachrichtendeskriptor (`MQMD`) festlegen oder Werte im `MQMD` abfragen. Mit dem zusätzlichen Entitätsheader `x-msg-usr` werden Informationen zu Benutzereigenschaften festgelegt und zurückgegeben, die Sie mit einer Anforderung verbinden möchten.

Entitätsheader können in unterschiedlichen HTTP-Anforderungskontexten verwendet werden:

DELETE
: Die Entitätsheader `x-msg-correlId` und `x-msg-msgId` können jeweils allein oder zusammen in der HTTP-Anforderung **DELETE** verwendet werden. Die Header bewirken, dass eine bestimmte Nachricht nach `MsgId` und `CorrelId` in einem `MQGET`-Aufruf ausgewählt und in der Warteschlange gelöscht wird.

GET
: Die Entitätsheader `x-msg-correlId` und `x-msg-msgId` können jeweils allein oder zusammen in der HTTP-Anforderung **GET** verwendet werden. Die Header bewirken, dass eine bestimmte Nachricht nach `MsgId` und `CorrelId` in einem `MQGET`-Aufruf zum Durchsuchen ausgewählt wird.

POST
: In einer HTTP-Anforderung vom Typ **POST** können außer `x-msg-timestamp` alle Entitätsheader verwendet werden.

x-msg-require-headers
: In HTTP-Anforderungen vom Typ **GET**, **POST** oder **DELETE** können mehrere Entitätsheader im `x-msg-require-headers`-Anforderungsheader durch Kommas voneinander getrennt verwendet werden. Auf diese Weise werden die angegebenen Entitätsheader in der HTTP-Antwortnachricht zurückgegeben und enthalten den Wert der zugehörigen Nachrichteneigenschaft.

In der Beschreibung der einzelnen Header wird erläutert, in welchen Kontexten der Header von WebSphere MQ Bridge for HTTP verarbeitet wird. Im Header **POST**, `x-msg-require-headers`, wird der Header beispielsweise von WebSphere MQ Bridge for HTTP in einer HTTP-Anforderung vom Typ **POST** oder im Anforderungsheader `x-msg-require-headers` in einer HTTP-Anforderung vom Typ **POST**, **GET**oder **DELETE** verarbeitet. Wenn sich der Header in einem Kontext befindet, für den er nicht zulässig ist, wird er ignoriert. Es wird kein Fehler gemeldet.

Sie können jeden HTTP-Standardheader zur Verarbeitung durch den Web-Server oder einen anderen Anforderungshandler in Anforderungen einfügen. Entsprechend kann die Antwort andere HTTP-Standardheader enthalten, die vom Web-Server oder von einem anderen Antworthandler eingefügt wurden.

Benutzerdefinierte HTTP-Anforderungsheader

Die drei benutzerdefinierten HTTP-Anforderungsheader `x-msg-range`, `x-msg-require-headers` und `x-msg-wait` übergeben zusätzliche Informationen zur HTTP-Anforderung an den Server. Sie dienen als Anforderungswerte. Mit `x-msg-range` können Sie die Anzahl der in einer Antwort zurückgegebenen Nachrichtendaten beschränken. Mit `x-msg-require-headers` können Sie festlegen, dass die Antwort Informationen zum Ergebnis der Anforderungen enthält. Mit `x-msg-wait` können Sie die Zeit ändern, die der Client auf eine HTTP-Antwort wartet.

HTTP-Standardheader

Der HTTP-Standardanforderungsheader `Host` muss in einer HTTP/1.1-Anforderung angegeben werden.

Die HTTP-Standardentitätsheader `Content-Length` und `Content-Type` können in einer Anforderung angegeben werden.

Die HTTP-Standardentitätsheader `Content-Length`, `Content-Location`, `Content-Range`, `Content-Type` und `Server` können als Antwort auf eine Anforderung zurückgegeben werden. Geben Sie einen oder mehrere HTTP-Standarddeader im `x-msg-request-header`-Header in der Anforderungsnachricht an.

Alphabetische Liste mit HTTP-Headern

`class`: HTTP-Entitätsheader `x-msg-class`

Legt den Nachrichtentyp fest oder gibt ihn zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-class`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	POST, `x-msg-require-headers`
Zulässige Werte	**BYTES** **MAP** **STREAM** **TEXT**
Standardwert	BYTES

Beschreibung

- Legt in einer HTTP-Anforderung vom Typ **POST** den Typ der erstellten Nachricht fest.
- Wenn Sie den Klassenheader in **GET** oder **DELETE** angeben, wird ein `400 Bad Request` mit dem Entitätshauptteil `MQHTTP40007`zurückgegeben.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-class` in der HTTP-Antwortnachricht auf den Typ einer Nachricht festgelegt.
- Wenn für diesen Header ein ungültiger Wert angegeben wird, wird eine `MQHTTP40005`-Nachricht zurückgegeben.
- Wenn der `x-msg-class`-Header nicht angegeben wird und der Inhaltstyp der Nachricht `application/x-www-form-urlencoded` lautet, wird davon ausgegangen, dass es sich bei den Daten um ein JMS-Zuordnungsobjekt handelt.

`Content-Length`: HTTP-Entitätsheader

Legt die Länge des Nachrichtentexts in Bytes fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`Content-Length`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	`x-msg-require-headers`
Zulässiger und zurückgegebener Wert	***`Integer value`*** Länge des Nachrichtentexts in Bytes.

Beschreibung

- Der `Content-Length`-Header ist in einer HTTP-Anforderung optional. Bei einer Anforderung vom Typ **GET** oder **DELETE** muss die Länge null betragen. Wenn bei Anforderungen vom Typ **POST** der `Content-`

Length-Header angegeben wird und nicht mit der Länge der Nachrichtenzeile übereinstimmt, wird die Nachricht abgeschnitten oder mit Nullen aufgefüllt, bis sie der angegebenen Länge entspricht.

- Der Content-Length-Header wird immer in der HTTP-Antwort zurückgegeben, auch wenn kein Inhalt vorhanden ist. In diesem Fall beträgt der Wert null.

Content-Location: HTTP-Entitätsheader

Gibt die Warteschlange oder das Thema, die bzw. das in der Anforderung angegeben ist, im Standardheader Content-Location in der HTTP-Antwortnachricht zurück.

Typ	Bezeichnung
Name des HTTP-Headers	Content-Location
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	x-msg-require-headers
Rückgabewert	URI im Format */msg/queue/queuename* oder */msg/topic/topicname*

Beschreibung

- Wenn in x-msg-require-headers angefordert, gibt der Entitätsheader Content-Location die Warteschlange oder das Thema zurück, die bzw. das in der HTTP-Anforderung angegeben ist.

Content-Range: HTTP-Entitätsheader

Gibt den in einer WebSphere MQ-Nachricht im Content-Range-Header in einer HTTP-Antwort ausgewählten Bytebereich zurück.

Typ	Bezeichnung
Name des HTTP-Headers	Content-Range
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	x-msg-require-headers
Rückgabewert	***String*** Gibt die Untergrenze *m* und die Obergrenze *n* der zurückgegebenen Unterzeichenfolge und *length* der gesamten Nachricht zurück. Zum Beispiel: *m-n/length*

Beschreibung

-
- Der Header Content-Range wird nur in der HTTP-Antwort zurückgegeben, wenn Content-Range in einer Anforderung vom Typ **GET** oder **DELETE** angegeben wird, die einen x-msg-range-Anforderungsheader enthält.
- Wenn x-msg-range in einer Anforderung vom Typ **GET** oder **DELETE** angegeben wird, wird der im "Content-Range"-Header angegebene Bytebereich in der Antwort zurückgegeben. Beispiel: Wenn x-

`msg-range: 0-60` in einer Anforderung für eine Nachricht mit `100` Bytes verwendet wird, enthält der Header des Inhaltsbereichs die Zeichenfolge `0-60/100`.

- Eine `x-msg-range`-Anforderung gibt zudem den Inhaltsbereich im `x-msg-range`-Header in der HTTP-Antwort zurück.

Content-Type: HTTP-Entitätsheader

Legt die Klasse der JMS-Nachricht in einer WebSphere MQ-Nachricht gemäß dem HTTP-Inhaltstyp fest oder ruft sie ab.

Typ	Bezeichnung
Name des HTTP-Headers	`Content-Type`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`
Zulässiger oder zurückgegebener Wert	***media-type*** Informationen zu unterstützten Datenträgertypen finden Sie in Tabelle 601 auf Seite 1255.

Tabelle 601. Zuordnung zwischen `x-msg-class` und HTTP `Content-Type`

x-msg-class	HTTP Content-Type
BYTES	`application/octet-stream` `application/xml`
TEXT	`text/*`
MAP	`application/x-www-form-urlencoded` `application/xml` (optional)
STREAM	`application/xml` (optional)

Beschreibung

- In einer HTTP-Anforderung vom Typ **POST** wird entweder der `Content-Type`-Header oder der `x-msg-class`-Header angegeben. Wenn Sie beide angeben, müssen sie konsistent sein oder es wird eine `HTTP Bad Request` Ausnahmebedingung `Status code 400` zurückgegeben. Wenn keiner der beiden Header `Content-Type` und `x-msg-class` angegeben wird, wird davon ausgegangen, dass `Content-Type` den Wert `text/*` aufweist.
- Der `Content-Type`-Header wird in der Antwort auf eine HTTP-Anforderung vom Typ **GET** oder **DELETE**, die einen Nachrichtentext enthält, immer festgelegt. Der `Content-Type`-Header wird gemäß den Regeln in Tabelle 602 auf Seite 1255 festgelegt.

Tabelle 602. Nachrichtentypen zu `x-msg-class` und `Content-Type` zuordnen

Nachrichtenformat	JMS-Nachrichtentyp	x-msg-class	Content-Type
Alles außer `MQFMT_STRING`	--	BYTES	`application/octet-stream`
`MQFMT_STRING`	--	TEXT	`text/plain`

Tabelle 602. Nachrichtentypen zu `x-msg-class` und `Content-Type` zuordnen (Forts.)			
Nachrichtenformat	**JMS-Nachrich-tentyp**	**`x-msg-class`**	**`Content-Type`**
`MQFMT_NONE`	`jms_bytes`	`BYTES`	`application/octet-stream`
`MQFMT_NONE`	`jms_text`	`TEXT`	`text/plain`
`MQFMT_NONE`	`jms_map`	`MAP`	`application/xml`
`MQFMT_NONE`	`jms_stream`	`STREAM`	`application/xml`

`correlId`: HTTP-Entitätsheader `x-msg-correlId`

Legt die Korrelations-ID fest oder gibt sie zurück.

Typ	**Bezeichnung**
Name des HTTP-Headers	`x-msg-correlId`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**DELETE**, **GET**, **POST**, `x-msg-require-headers`

Zulässige Werte	***`String value`***
	Beispiel:
	`x-msg-correlId: mycorrelationid`
	Zeichenfolgen in Anführungszeichen sind zulässig. Beispiel:
	`x-msg-correlId: "my id"`
	`Hex value`
	Ein Hexadezimalwert mit vorangestelltem `0x:`. Beispiel:
	`x-msg-correlId: 0x:43c1d23a`
	Der Hexadezimalwert nach 0x: ist auf 48 Zeichen begrenzt, die 24 Byte darstellen. Zusätzliche Daten werden ignoriert.

Standardwert	Nicht zutreffend

Beschreibung

- Legt in einer HTTP-Anforderung vom Typ **POST** die Korrelations-ID der erstellten Nachricht fest.
- Wählt in einer HTTP-Anforderung vom Typ **GET** oder **DELETE** die Nachricht in der Warteschlange oder im Thema aus. Wenn keine Nachricht mit der angegebenen Korrelations-ID vorhanden ist, wird eine `HTTP 504 Gateway Timeout` -Antwort zurückgegeben. `x-msg-correlId` kann mit `x-msg-msgID` verwendet werden, um eine Nachricht in einer Warteschlange oder in einem Thema auszuwählen, die mit beiden Selektoren übereinstimmt.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-corelId` in der HTTP-Antwortnachricht auf die Korrelations-ID einer Nachricht festgelegt.
- Horizontale Leerzeichen nach dem Präfix `0x:` sind zulässig.

Anmerkung:

- Wenn `x-msg-correlId` in einer HTTP-Anforderung vom Typ **GET** oder **DELETE** ohne einen Wert angegeben wird (Beispiel: "`x-msg-correlId:`"), wird unabhängig von der jeweiligen Korrelations-ID die nächste Nachricht in der Warteschlange oder im Thema zurückgegeben.
- Wenn Sie einen Selektor mit maximal 24 Zeichen oder 0x: gefolgt von maximal 48 Zeichen eingeben, verwendet WebSphere MQ Bridge for HTTP einen optimierten Selektor zur Verbesserung der Leistung.
- Ein JMS-Nachrichtenselektor, der JMSCorrelationID enthält, wird bei der Auswahl von Nachrichten in der Warteschlange verwendet. Dieser Selektor verhält sich wie im Abschnitt Auswahlverhalten beschrieben.

encoding: HTTP-Entitätsheader `x-msg-encoding`

Legt die Nachrichtencodierung fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-encoding`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`
Zulässige Werte	Eine durch Kommas getrennte Liste mit folgenden Werten: **DECIMAL_NORMAL** **DECIMAL_REVERSED** **FLOAT_IEEE_NORMAL** **FLOAT_IEEE_REVERSED** **FLOAT_S390** **INTEGER_NORMAL** **INTEGER_REVERSED** Zum Beispiel: `x-msg-encoding: INTEGER_NORMAL,DECIMAL_NORMAL,FLOAT_IEEE_NORMAL` **Anmerkung:** Bei dem Wert muss die Groß-/Kleinschreibung beachtet werden.
Standardwert	`DECIMAL_NORMAL, FLOAT_IEEE_NORMAL, INTEGER_NORMAL`

Beschreibung

- Legt in einer HTTP-Anforderung vom Typ **POST** die Codierung der erstellten Nachricht fest.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-encoding`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-encoding` in der HTTP-Antwort auf die Codierungseigenschaft einer Nachricht festgelegt.

expiry: HTTP-Entitätsheader `x-msg-expiry`

Legt die Dauer bis zum Ablauf der Nachricht fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-expiry`
Typ des HTTP-Headers	Entitätsheader

Typ	Bezeichnung
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	**UNLIMITED**
	Beispiel:
	`x-msg-expiry: UNLIMITED`
	Integer value
	Millisekunden vor Ablauf.
	Beispiel:
	`x-msg-expiry: 10000`

Standardwert	UNLIMITED

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, läuft die Nachricht in der angegebenen Zeit ab.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-expiry`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-expiry` in der HTTP-Antwortnachricht auf die Ablaufzeit einer Nachricht festgelegt.
- UNLIMITED gibt an, dass die Nachricht nie abläuft.
- Die Ablaufzeit einer Nachricht beginnt zu dem Zeitpunkt, an dem die Nachricht in der Warteschlange ankommt. Daher spielt die Netzlatenz keine Rolle.
- Der Maximalwert wird durch WebSphere MQ auf 214748364700 Millisekunden begrenzt. Wenn ein größerer Wert angegeben wird, wird von der maximal möglichen Ablaufzeit ausgegangen.

`format`: HTTP-Entitätsheader `x-msg-format`

Legt das WebSphere MQ-Nachrichtenformat fest oder gibt es zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-format`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	**NONE**
	Zum Beispiel:
	`x-msg-format: NONE`
	String value
	Ein benutzerdefinierter Wert mit bis zu acht Zeichen. Zum Beispiel:
	`x-msg-format: myformat`

Standardwert	None

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, wird das Format der Anforderungsnachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-format`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-format` in der HTTP-Antwortnachricht auf das Format einer Nachricht festgelegt.
- Bei `NONE` muss die Groß-/Kleinschreibung beachtet werden. Zudem wird damit angegeben, dass das Nachrichtenformat leer ist.
- Der Wert von `x-msg-format` wird verwendet, auch wenn er im Widerspruch zum Medientyp der HTTP-Anforderung steht. Weitere Informationen hierzu finden Sie im Abschnitt Tabelle 603 auf Seite 1259.

Tabelle 603. Inhaltstyp und `x-msg-class` zum Nachrichtenformat zuordnen

x-msg-class	**Inhaltstyp**	**Nachrichtenformat in Warteschlange/Thema**
BYTES	• `application/octet-stream` • application/xml	WebSphere MQ-Nachricht: `MQFMT` festgelegt auf `MQC.MQFMT_NONE`
TEXT	• `text/*`	WebSphere MQ-Nachricht: `MQFMT` festgelegt auf `MQC.MQFMT_STRING`
MAP	• `application/x-www-form-urlencoded` • `application/xml` (optional)	`JMSMap`
STREAM	• `application/xml` (optional)	`JMSStream`

msgId: HTTP-Entitätsheader `x-msg-msgId`

Legt die Nachrichten-ID fest oder gibt sie zurück.

Typ	**Bezeichnung**
Name des HTTP-Headers	`x-msg-msgId`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**DELETE**, **GET**, **POST**, `x-msg-require-headers`

Zulässige Werte	***String value***
	Zum Beispiel:
	`x-msg-msgId: mymsgid`
	Zeichenfolgen in Anführungszeichen, z. B. `x-msg-msgId: "my id"`
	Hex value
	Ein Hexadezimalwert mit vorangestelltem `0x:`. Beispiel:
	`x-msg-msgId: 0x:43c1d23a`

Standardwert	Nicht zutreffend

Beschreibung

- Legt in einer HTTP-Anforderung vom Typ **POST** die Nachrichten-ID der erstellten Nachricht fest.
- Wählt in einer HTTP-Anforderung vom Typ **GET** oder **DELETE** die Nachricht in der Warteschlange oder im Thema aus. Wenn keine Nachricht mit der angegebenen Nachrichten-ID vorhanden ist, wird die Antwort `HTTP 504 Gateway Timeout` zurückgegeben. `x-msg-msgId` kann mit `x-msg-correlID` verwendet werden, um eine Nachricht in einer Warteschlange oder in einem Thema auszuwählen, die mit beiden Selektoren übereinstimmt.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-msgId` in der HTTP-Antwort auf die Nachrichten-ID einer Nachricht zurückgegeben.
- Horizontale Leerzeichen nach dem Präfix `0x:` sind zulässig.

Anmerkung: Wenn `x-msg-msgId` in einer HTTP-Anforderung vom Typ **GET** oder **DELETE** ohne einen Wert angegeben wird (Beispiel: "`x-msg-msgId:`"), wird unabhängig von der jeweiligen Nachrichten-ID die nächste Nachricht in der Warteschlange oder im Thema zurückgegeben.

Ein JMS-Nachrichtenselektor, der JMSMessageID enthält, wird bei der Auswahl von Nachrichten in der Warteschlange verwendet. Dieser Selektor verhält sich wie im Abschnitt Auswahlverhalten beschrieben.

`persistence`: HTTP-Entitätsheader `x-msg-persistence`

Legt die Nachrichtenpersistenz fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-persistence`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	**NON_PERSISTENT**
	Die Nachricht geht bei Systemausfällen und Neustarts des Warteschlangenmanagers verloren.
	Zum Beispiel:
	`x-msg-persistence: NON_PERSISTENT`
	PERSISTENT
	Die Nachricht wird bei Systemausfällen und Neustarts des Warteschlangenmanagers beibehalten.
	Zum Beispiel:
	`x-msg-persistence: PERSISTENT`
	AS_DESTINATION
	Gilt nur für **POST**.
	Verwendet die Standardpersistenz des Ziels wie vom Messaging-Provider festgelegt.
	Anmerkung: Groß-/Kleinschreibung muss beachtet werden

Standardwert	NON_PERSISTENT

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, wird die Persistenz der Anforderungsnachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-persistence`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-persistence` in der HTTP-Antwortnachricht auf die Persistenz einer Nachricht festgelegt.

`priority`: HTTP-Entitätsheader `x-msg-priority`

Legt die Priorität der Nachricht fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-priority`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	**LOW**
	Zum Beispiel:
	`x-msg-priority: LOW`
	MEDIUM
	Diese Priorität entspricht der WebSphere MQ -Prioritätsstufe 4. Zum Beispiel:
	`x-msg-priority: MEDIUM`
	HIGH
	Zum Beispiel:
	`x-msg-priority: HIGH`
	Integer value
	Eine Zeichenfolgedarstellung einer Ganzzahl im Bereich zwischen 0 und 9. Beispiel:
	`x-msg-priority: 3`
	AS_DESTINATION
	Gilt nur für **POST**.
	Verwendet die Standardpriorität des Ziels wie vom Messaging-Provider festgelegt.
	Anmerkung: Groß-/Kleinschreibung muss beachtet werden

Standardwert	`MEDIUM`

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, wird die Priorität der Anforderungsnachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-priority`-Header ignoriert.

- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-priority` in der HTTP-Antwortnachricht auf die Priorität einer Nachricht festgelegt.

`priority`: HTTP-Entitätsheader `x-msg-priority`

Legt die Priorität der Nachricht fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-priority`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	**LOW**
	Zum Beispiel:
	`x-msg-priority: LOW`
	MEDIUM
	Diese Priorität entspricht der WebSphere MQ -Prioritätsstufe 4. Zum Beispiel:
	`x-msg-priority: MEDIUM`
	HIGH
	Zum Beispiel:
	`x-msg-priority: HIGH`
	Integer value
	Eine Zeichenfolgedarstellung einer Ganzzahl im Bereich zwischen 0 und 9. Beispiel:
	`x-msg-priority: 3`
	AS_DESTINATION
	Gilt nur für **POST**.
	Verwendet die Standardpriorität des Ziels wie vom Messaging-Provider festgelegt.
	Anmerkung: Groß-/Kleinschreibung muss beachtet werden

Standardwert	`MEDIUM`

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, wird die Priorität der Anforderungsnachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-priority`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-priority` in der HTTP-Antwortnachricht auf die Priorität einer Nachricht festgelegt.

`replyTo`: HTTP-Entitätsheader `x-msg-replyTo`

Legt die Warteschlange für zu beantwortende Nachrichten und den Warteschlangenmanagernamen fest oder gibt ihn zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-replyTo`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungs-nachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	***URI***
	Ein Punkt-zu-Punkt-URI. Beispiel:
	`x-msg-replyTo: /msg/queue/myReplyQueue`
	`x-msg-replyTo: /msg/queue/myReplyQueue@myReplyQueueManager`
	Anmerkung: Groß-/Kleinschreibung muss beachtet werden

Standardwert	`MEDIUM`

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, wird das "replyTo"-Ziel der Anforderungs-nachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-replyTo`-Header ignoriert.
- Wird in `x-msg-require-headers`angegeben, wird `x-msg-replyTo` in der HTTP-Antwortnachricht auf die Empfangswarteschlange für Antworten und den Warteschlangenmanagernamen einer -Nach-richt gesetzt.

Anmerkung: Der URI in der HTTP-Antwort kann den Namen des Warteschlangenmanagers enthalten, mit dem die WebSphere MQ -Bridge für HTTP verbunden ist.

`Server`: HTTP-Antwortheader

Gibt Informationen zum Server und Protokoll zurück, mit dem der Client verbunden ist.

Typ	Bezeichnung
Name des HTTP-Headers	`Server`
Typ des HTTP-Headers	Antwortheader
Gültig in HTTP-Anforderungsnachricht	`x-msg-require-headers`

Rückgabewert	`WMQ-HTTP/1.1 JEE-Bridge/1.1`
	oder
	`Server: Product-token WMQ-HTTP/1.1 JEE-Bridge/1.1`

Beschreibung

- Wenn WebSphere MQ Bridge for HTTP auf einem Anwendungsserver implementiert wird, werden die Details zu WebSphere MQ Bridge for HTTP zum "server"-Antwortheader hinzugefügt. Beispiel: Web-Sphere MQ Bridge for HTTP implementiert unter WebSphere Application Server Community Edition, mit der Bezeichnung Apache-Coyote, gibt folgende Antwort:

```
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
```

`require-headers`: HTTP-Anforderungsheader `x-msg-require-headers`

Legt fest, welche Header in der HTTP-Antwortnachricht zurückgegeben werden.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-require-headers`
Typ des HTTP-Headers	Anforderungsheader
Gültig in HTTP-Anforderungs-nachricht	**POST**, **GET**, **DELETE**

Zulässige Werte	Eine durch Kommas getrennte Liste mit den Namen der Entitätsheader:
	ALL
	ALL-USR
	class
	content-location
	correlId
	encoding
	expiry
	format
	msgId
	NO_require-headers
	persistence
	priority
	replyTo
	server
	timestamp
	usr-*property name*
	Zum Beispiel:
	`x-msg-require-headers: msgId`
	oder
	`x-msg-require-headers: expiry,correlId,timestamp`
	So wird eine bestimmte Eigenschaft angefordert:
	`x-msg-require-headers: usr-myCustomProperty`
	So werden alle Eigenschaften angefordert:
	`x-msg-require-headers: ALL-USR, ALL`

Standardwert	`NO_require-headers`

Beschreibung

- Beim Wert von `x-msg-require-headers` muss die Groß-/Kleinschreibung außer bei den Konstanten `ALL`, `NO_require-headers` und `ALL-USR` und der Variablen *`property-name`* nicht beachtet werden.

`timestamp`: HTTP-Entitätsheader `x-msg-timestamp`

Gibt die Zeitmarke der Nachricht zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-timestamp`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungs-nachricht	`x-msg-require-headers`
Rückgabewert	***HTTP-date*** Ein Datum im Format "Wochentag, Tag Monat Jahr Uhrzeit Zeitzone". Beispiel: `Sun, 06 Nov 1994 08:49:37 GMT` Definiert durch RFC 822 und aktualisiert in RFC 1123.
Standardwert	Nicht zutreffend

Beschreibung

- In einer HTTP-Anforderung vom Typ **POST**, **GET** oder **DELETE** wird der `x-msg-timestamp`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-timestamp` in der HTTP-Antwortnachricht auf die Zeitmarke einer Nachricht festgelegt.

`usr`: HTTP-Entitätsheader `x-msg-usr`

Legt die Benutzereigenschaften fest oder gibt sie zurück.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-usr`
Typ des HTTP-Headers	Entitätsheader
Gültig in HTTP-Anforderungsnachricht	**POST**, `x-msg-require-headers`

Zulässige Werte	Informationen hierzu finden Sie im Abschnitt „Syntax” auf Seite 1266. Beispiel:
	`x-msg-usr: myProp1;5;i1,`
	`x-msg-usr: myProp2;"My String";string`

Standardwert	Nicht zutreffend

Beschreibung

- Wenn in einer HTTP-Anforderung vom Typ **POST** festgelegt, werden die Benutzereigenschaften der Anforderungsnachricht festgelegt.
- In einer HTTP-Anforderung vom Typ **GET** oder **DELETE** wird der `x-msg-usr`-Header ignoriert.
- Wenn in `x-msg-require-headers` angegeben, wird `x-msg-usr` in der HTTP-Antwortnachricht auf die Benutzereigenschaften einer Nachricht festgelegt.
- In einer Nachricht können mehrere Eigenschaften festgelegt werden. Mehrere Eigenschaften werden durch Kommas getrennt in einem `x-msg-usr`-Header oder mithilfe von mehreren separaten Instanzen des `x-msg-usr`-Headers angegeben.

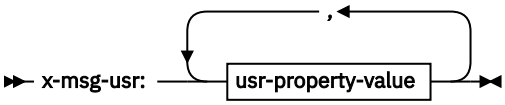
- Sie können anfordern, dass in der Antwort auf eine Anforderung vom Typ **GET** oder **DELETE** eine bestimmte Eigenschaft zurückgegeben wird. Geben Sie den Namen der Eigenschaft im `x-msg-require-headers`-Header der Anforderung mithilfe des Präfixes `usr-` an. Zum Beispiel:

```
x-msg-require-headers: usr-myProp1
```

- Um anzufordern, dass alle Benutzereigenschaften in einer Antwort zurückgegeben werden, verwenden Sie die Konstante `ALL-USR`. Zum Beispiel:

```
x-msg-require-headers: ALL-USR
```

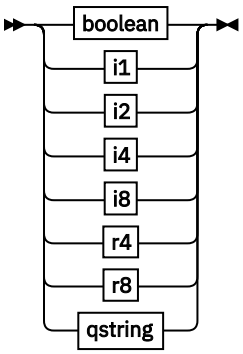
Syntax



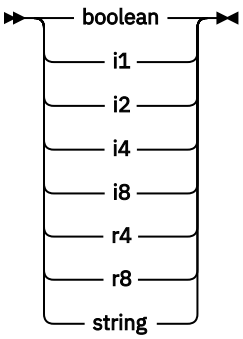
usr-property-value

property-name

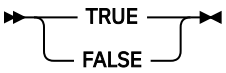
usr-value



usr-type



boolean



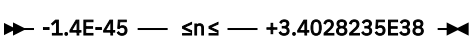
i1

i2

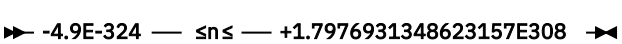
i4

i8

r4



r8



qstring

`wait`: HTTP-Anforderungsheader `x-msg-wait`

Legen Sie fest, wie lange auf den Eingang einer Nachricht gewartet wird, bevor eine `HTTP 504 Gateway Timeout` -Antwortnachricht zurückgegeben wird.

Typ	Bezeichnung
Name des HTTP-Headers	`x-msg-wait`
Typ des HTTP-Headers	Anforderungsheader
Gültig in HTTP-Anforderungs-nachricht	**GET**, **DELETE**

Zulässiger Wert	**NO_WAIT**
	Zum Beispiel:
	`x-msg-wait: NO_WAIT`
	Integer value
	Die Anzahl der Millisekunden, die WebSphere MQ Bridge for HTTP auf die Ankunft einer Nachricht wartet. Beispiel:
	`x-msg-wait: 8`

Standardwert	`NO_WAIT`

Beschreibung

- In einer HTTP-Anforderung vom Typ **POST** wird der `x-msg-wait`-Header ignoriert.
- In einer HTTP-Anforderung **GET** oder **DELETE** gibt `x-msg-wait` an, wie lange auf den Eingang einer Nachricht gewartet werden soll, bevor eine `HTTP 504 Gateway Timeout` -Antwort zurückgegeben wird.
- Bei `NO_WAIT` muss die Groß-/Kleinschreibung beachtet werden.
- Die Standardeinstellung für die maximale Wartezeit beträgt 35000. Sie können die Standardeinstellung ändern, indem Sie den `maximum_wait_time`-Parameter des Servlets ändern. Weitere Informationen finden Sie im Abschnitt WebSphere MQ Bridge for HTTP installieren, konfigurieren und überprüfen.
- Wenn Sie einen Wert festlegen, der größer als `maximum_wait_time` ist, wird stattdessen `maximum_wait_time` verwendet.

HTTP-Rückkehrcodes

Liste mit Rückkehrcodes aus WebSphere MQ Bridge for HTTP

WebSphere MQ Bridge for HTTP gibt vier Fehlertypen zurück:

Servletfehler
MQHTTP0001 und MQHTTP0002 sind Servletfehler. Sie werden protokolliert, aber nicht an den HTTP-Client zurückgegeben.

Erfolgreiche Vorgänge
Ein HTTP-Statuscode im Bereich von 200 bis 299 gibt einen erfolgreichen Vorgang an.

Clientfehler
Ein HTTP-Statuscode im Bereich von 400 bis 499 gibt einen Clientfehler an. WebSphere MQ Bridge for HTTP-Rückkehrcodes im Bereich von MQHTTP40001 bis MQHTTP49999 entsprechen Clientfehlern.

Serverfehler

Ein HTTP-Statuscode im Bereich von 500 bis 599 gibt einen Serverfehler an. WebSphere MQ Bridge for HTTP-Rückkehrcodes im Bereich von MQHTTP50001 bis MQHTTP59999 entsprechen Serverfehlern.

Wenn ein Serverfehler auftritt, wird im Fehlerprotokoll des Anwendungsservers ein vollständiger Stack-Trace ausgebeben. Der Stack-Trace wird auch an den HTTP-Client in der HTTP-Antwort zurückgegeben. Verarbeiten Sie den Stack-Trace in der Clientanwendung oder wenden Sie sich damit an den Administrator des Anwendungsservers, um den Fehler zu beheben.

Wenn der Stack-Trace Fehler des Ressourcenadapters enthält, finden Sie entsprechende Informationen in der Dokumentation zu Ihrem Ressourcenadapter.

Alphabetische Liste mit Rückkehrcodes

HTTP 200: OK

Diese Statuscodeklasse gibt an, dass die Anforderung erfolgreich eingegangen ist sowie verstanden und akzeptiert wurde.

HTTP-Statuscode
`200 OK`

HTTP 204: Kein Inhalt

Gesendet nach einem erfolgreichen HTTP **GET** oder **DELETE** und `x-msg-range: 0` wurde in der Anforderung gesendet.

HTTP-Statuscode
`204 No Content`

MQHTTP0001: Keine Verbindungsfactory angegeben im Servlet-Kontext.

Servletfehler

Beschreibung
Servletfehler

HTTP-Statuscode
--

Programmiereraktion
Wo diese Fehler protokolliert werden, richtet sich nach dem Anwendungsserver. Weitere Informationen finden Sie in der Dokumentation zu Ihrem Anwendungsserver.

MQHTTP0002: Verbindungsmanager für *queueOrTopic* konnte nicht mit dem JNDI-Namen von *jndiNameTried* abgerufen werden.

Servletfehler

Beschreibung
Servletfehler

HTTP-Statuscode
--

Programmiereraktion
Wo diese Fehler protokolliert werden, richtet sich nach dem Anwendungsserver. Weitere Informationen finden Sie in der Dokumentation zu Ihrem Anwendungsserver.

MQHTTP40001: Reserviert

Reserved

HTTP-Statuscode
`400 Bad Request`

MQHTTP40002: URI ist nicht gültig für WebSphere MQ Transport for HTTP

Der in der HTTP-Anforderung angegebene URI ist ungültig.

Beschreibung
Der in der HTTP-Anforderung angegebene URI ist ungültig.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Bestätigen Sie, dass das Format und die Syntax des angegebenen URI ordnungsgemäß sind.

MQHTTP40003: URI ist ungültig. `@qmgr` ist nur bei POST-Anforderungen gültig.

Die URI-Option `@qmgr` wurde in einem URI für eine HTTP-Anforderung angegeben, bei der es sich nicht um eine Anforderung vom Typ **POST** handelt.

Beschreibung
Die URI-Option `@qmgr` wurde in einem URI für eine HTTP-Anforderung angegeben, bei der es sich nicht um eine Anforderung vom Typ **POST** handelt.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Wenn Sie eine Nachricht mithilfe des Verbs **POST** in eine Warteschlange stellen möchten, verwenden Sie anstelle der HTTP-Anforderung eine Anforderung vom Typ **POST**. Wenn Sie eine Nachricht mithilfe der Verben **DELETE** oder **GET** abrufen möchten, entfernen Sie `@qmgr` aus dem URI.

MQHTTP40004: Ungültiger `Content-Type`-Header angegeben

Das in einer Anforderung vom Typ **POST** angegebene `Content-Type`-Headerfeld ist mit dem `x-msg-class`-Headerwert nicht kompatibel.

Beschreibung

Das in einer Anforderung vom Typ **POST** angegebene `Content-Type`-Headerfeld ist mit dem `x-msg-class`-Headerwert nicht kompatibel.

HTTP-Statuscode

`400 Bad Request`

Programmiereraktion

Verwenden Sie anstelle des `Content-Type`-Headerfelds ein Headerfeld, das unterstützt wird. Der `Content-Type`-Header muss mit dem angegebenen `x-msg-class`-Headerfeld kompatibel sein.

MQHTTP40005: Falscher Nachrichtenheaderwert

Ein unterstütztes Headerfeld wurde mit einem Wert angegeben, der für die angegebene Anforderung nicht gültig ist.

Beschreibung

Ein unterstütztes Headerfeld wurde mit einem Wert angegeben, der für die angegebene Anforderung nicht gültig ist.

HTTP-Statuscode

`400 Bad Request`

Programmiereraktion

Verwenden Sie anstelle des für das angegebene Headerfeld festgelegten Werts einen gültigen Wert. Überprüfen Sie die Groß-/Kleinschreibung des festgelegten Werts, da bei den Werten einiger Headerfelder die Groß-/Kleinschreibung beachtet werden muss.

MQHTTP40006: *`Header_name`* ist kein gültiger Anforderungsheader

In einer HTTP-Anforderungsnachricht wurde ein Header angegeben, der nur in einer HTTP-Antwortnachricht gültig ist.

Beschreibung

In einer HTTP-Anforderungsnachricht wurde ein Header angegeben, der nur in einer HTTP-Antwortnachricht gültig ist.

HTTP-Statuscode

`400 Bad Request`

Programmiereraktion

Entfernen Sie Header aus der HTTP-Anforderung, die nur in einer HTTP-Antwort gültig sind. Beispiel: `x-msg-timestamp`.

MQHTTP40007: *`Header_name`* ist nur gültig auf ...

Ein Header wurde in einer HTTP-Anforderung festgelegt, aber das Headerfeld ist für das angegebene Anforderungsverb nicht gültig.

Beschreibung
Ein Header wurde in einer HTTP-Anforderung festgelegt, aber das Headerfeld ist für das angegebene Anforderungsverb nicht gültig.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Entfernen Sie Header aus der HTTP-Anforderung, die für das angegebene Anforderungsverb nicht gültig sind. Beispiel: `x-msg-encoding` ist gültig für HTTP-Anforderungen vom Typ **POST**, aber nicht gültig für HTTP-Anforderungen vom Typ **GET** oder **DELETE**.

MQHTTP40008: *`Header_name`* Maximale Länge ist ...

Die maximale Länge für das angegebene Headerfeld wurde überschritten.

Beschreibung
Die maximale Länge für das angegebene Headerfeld wurde überschritten.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Verwenden Sie anstelle des Werts im Headerfeld einen Wert, der in dem für das Headerfeld zulässigen Bereich liegt.

MQHTTP40009: Headerfeld *`header_field`* ist ungültig für ...

Ein in einer HTTP-Anforderung angegebenes Headerfeld wird von dem Messaging-Provider, mit dem WebSphere MQ Bridge for HTTP verbunden ist, nicht unterstützt.

Beschreibung
Ein in einer HTTP-Anforderung angegebenes Headerfeld wird von dem Messaging-Provider, mit dem WebSphere MQ Bridge for HTTP verbunden ist, nicht unterstützt. Der Fehler tritt auf, wenn ein Messaging-Provider verwendet wird, der nicht alle Features von WebSphere MQ Bridge for HTTP unterstützt.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Entfernen Sie den nicht unterstützten Header aus der HTTP-Anforderung.

MQHTTP40010: Nachricht mit `Content-Type` *`content_type`* konnte nicht syntaktisch analysiert werden

Der Inhalt der HTTP-Anforderung ist mit dem `Content-Type`-Header der Anforderung nicht kompatibel.

Beschreibung
Der Inhalt der HTTP-Anforderung ist mit dem `Content-Type`-Header der Anforderung nicht kompatibel. Eine häufige Ursache ist `application/x-www-form-urlencoded` oder `application/xml data`.

HTTP-Statuscode
`400 Bad Request`

Programmiereraktion
Korrigieren Sie den Inhalt der HTTP-Anforderung, sodass er das richtige Format für den `Content-Type`-Header der Anforderung aufweist.

MQHTTP40301: Sie haben keine Berechtigung für den Zugriff auf ...

WebSphere MQ Bridge for HTTP konnte sich selbst nicht für das angegebene Ziel authentifizieren.

Beschreibung
WebSphere MQ Bridge for HTTP konnte sich selbst nicht für das angegebene Ziel authentifizieren.

HTTP-Statuscode
`403 Forbidden`

Programmiereraktion
Ändern Sie die Authentifizierungseigenschaften des Ziels so, dass WebSphere MQ Bridge for HTTP zum Herstellen einer Verbindung mit dem Ziel berechtigt ist. Oder geben Sie ein Ziel an, für das WebSphere MQ Bridge for HTTP über die Berechtigung zum Herstellen einer Verbindung verfügt.

MQHTTP40302: Sie haben keine Berechtigung für ...

WebSphere MQ Bridge for HTTP konnte keine Verbindung zum Warteschlangenmanager herstellen.

Beschreibung
WebSphere MQ Bridge for HTTP konnte keine Verbindung zum Warteschlangenmanager herstellen. Die Sicherheitskonfiguration von WebSphere MQ Bridge for HTTP ist falsch.

HTTP-Statuscode
`403 Forbidden`

Programmiereraktion
Ändern Sie die Authentifizierungskonfiguration des Warteschlangenmanagers so, dass WebSphere MQ Bridge for HTTP zum Herstellen einer Verbindung mit dem Warteschlangenmanager berechtigt ist. Oder konfigurieren Sie WebSphere MQ Bridge for HTTP für die Verbindung mit einem Warteschlangenmanager, für den das Produkt über die Berechtigung zum Herstellen einer Verbindung verfügt.

MQHTTP40401: Das Ziel *destination_name* konnte nicht gefunden werden

Das im URI der HTTP-Anforderung angegebene Ziel wurde von WebSphere MQ Bridge for HTTP nicht gefunden.

Beschreibung
Das im URI der HTTP-Anforderung angegebene Ziel wurde von WebSphere MQ Bridge for HTTP nicht gefunden.

HTTP-Statuscode
`404 Not found`

Programmiereraktion
Überprüfen Sie, ob das im URI der HTTP-Anforderung angegebene Ziel vorhanden ist, oder geben Sie ein anderes Ziel an.

MQHTTP40501: Methode *method_name* nicht zulässig

Die in der HTTP-Anforderung angegebene Methode wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

Beschreibung

Die in der HTTP-Anforderung angegebene Methode wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

HTTP-Statuscode

`405 Method not allowed`

Programmiereraktion

Verwenden Sie anstelle der in der HTTP-Anforderung angegebenen Methode eine Methode, die von WebSphere MQ Bridge for HTTP unterstützt wird.

MQHTTP41301: Die übergebene Nachricht war zu groß für das Ziel

Das im URI der HTTP-Anforderung vom Typ POST angegebene Ziel akzeptiert Nachrichten nicht, die so lang sind wie die in der HTTP-Anforderung.

Beschreibung

Das im URI der HTTP-Anforderung vom Typ POST angegebene Ziel akzeptiert Nachrichten nicht, die so lang sind wie die in der HTTP-Anforderung.

HTTP-Statuscode

`413 Request entity too large`

Programmiereraktion

Reduzieren Sie die Größe der in der HTTP-Anforderung angegebenen Nachricht. Oder geben Sie ein Ziel an, das Nachrichten mit der gewünschten Länge unterstützt.

MQHTTP41501: Der Zeichensatz des Datenträgertyps wird nicht unterstützt

Der im `Content-Type`-Headerfeld angegebene Zeichensatz wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

Beschreibung

Der im `Content-Type`-Headerfeld angegebene Zeichensatz wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

HTTP-Statuscode

`415 Unsupported media type`

Programmiereraktion

Verwenden Sie anstelle des Zeichensatzes im `Content-Type`-Headerfeld einen Zeichensatz, der von WebSphere MQ Bridge for HTTP unterstützt wird.

MQHTTP41502: Medientyp *media-type* wird nicht unterstützt ...

Der in der HTTP-Anforderung angegebene Datenträgertyp wird von WebSphere MQ Bridge for HTTP für das angegebene HTTP-Verb nicht unterstützt.

Beschreibung
Der in der HTTP-Anforderung angegebene Datenträgertyp wird von WebSphere MQ Bridge for HTTP für das angegebene HTTP-Verb nicht unterstützt.

HTTP-Statuscode
`415 Unsupported media type`

Programmiereraktion
Verwenden Sie anstelle des in der HTTP-Anforderung angegebenen Datenträgertyps einen Datenträgertyp, der von WebSphere MQ Bridge for HTTP für das angegebene HTTP-Verb unterstützt wird.

MQHTTP41503: Medientyp *media-type* wird nicht unterstützt ...

Der in der HTTP-Anforderung angegebene Datenträgertyp wird von WebSphere MQ Bridge for HTTP für das angegebene `x-msg-class`-Headerfeld nicht unterstützt.

Beschreibung
Der in der HTTP-Anforderung angegebene Datenträgertyp wird von WebSphere MQ Bridge for HTTP für das angegebene `x-msg-class`-Headerfeld nicht unterstützt.

HTTP-Statuscode
`415 Unsupported media type`

Programmiereraktion
Verwenden Sie anstelle des in der HTTP-Anforderung angegebenen Datenträgertyps einen Datenträgertyp, der von WebSphere MQ Bridge for HTTP für das angegebene `x-msg-class`-Headerfeld unterstützt wird.

MQHTTP41701: Der HTTP-Header `Expect` wird nicht unterstützt

WebSphere MQ Bridge for HTTP unterstützt das Headerfeld `Expect` nicht.

Beschreibung
In einer HTTP-Anforderung wurde der `Expect`-Header angegeben. WebSphere MQ Bridge for HTTP unterstützt das Headerfeld `Expect` nicht.

HTTP-Statuscode
`417 Expectation failed`

Programmiereraktion
Entfernen Sie den nicht unterstützten `Expect`-Header aus der HTTP-Anforderung.

MQHTTP50001: Es ist ein unerwarteter Fehler aufgetreten ...

Bei WebSphere MQ Bridge for HTTP ist ein Fehler aufgetreten.

Beschreibung
Bei WebSphere MQ Bridge for HTTP ist ein Fehler aufgetreten.

HTTP-Statuscode
`500 Internal server error`

Programmiereraktion
Wenden Sie sich an den Systemadministrator für WebSphere MQ Bridge for HTTP.

MQHTTP50201: Zwischen WebSphere MQ Bridge for HTTP und dem Warteschlangenmanager ist ein Fehler aufgetreten

Zwischen WebSphere MQ Bridge for HTTP und dem Warteschlangenmanager ist ein Fehler aufgetreten.

Beschreibung
Zwischen WebSphere MQ Bridge for HTTP und dem Warteschlangenmanager ist ein Fehler aufgetreten.

HTTP-Statuscode
`502 Bad Gateway`

Programmiereraktion
Wenden Sie sich an den Systemadministrator für WebSphere MQ Bridge for HTTP.

MQHTTP50401: Das zulässige Zeitlimit für den Nachrichtenabruf wurde überschritten

Im Zeitlimitintervall wurde keine Nachricht zurückgegeben, die den angegebenen Anforderungsparametern in einem HTTP **GET** oder HTTP **DELETE** entspricht.

Beschreibung
Im Zeitlimitintervall wurde keine Nachricht zurückgegeben, die den angegebenen Anforderungsparametern in einem HTTP **GET** oder HTTP **DELETE** entspricht. Der Rückkehrcode gibt an, dass zu keinem Zeitpunkt während der Lebensdauer der HTTP-Anforderung eine geeignete Nachricht vorhanden war.

HTTP-Statuscode
`504 Gateway timeout`

Programmiereraktion
Wenn eine Nachricht erwartet wurde, überprüfen Sie die Headerfelder der HTTP-Anforderung, wie etwa `x-msg-correlId` und `x-msg-msgid`. Stellen Sie sicher, dass das im URI der HTTP-Anforderung angegebene Ziel korrekt ist. Versuchen Sie, die Wartezeit der HTTP-Anforderung mithilfe des `x-msg-wait`-Headerfelds zu erweitern.

MQHTTP50501: HTTP 1.1 und höher ...

Das in der HTTP-Anforderung verwendete HTTP-Protokoll wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

Beschreibung
Das in der HTTP-Anforderung verwendete HTTP-Protokoll wird von WebSphere MQ Bridge for HTTP nicht unterstützt.

HTTP-Statuscode
`505 HTTP version not supported`

Programmiereraktion
Ändern Sie die HTTP-Anforderung so, dass HTTP-Protokoll V1.1 oder höher verwendet wird.

Nachrichtentypen und Nachrichtenzuordnungen für WebSphere Bridge for HTTP

WebSphere MQ Bridge for HTTP unterstützt die vier Nachrichtenklassen TEXT, BYTES, STREAM und MAP. Die Nachrichtenklassen werden JMS-Nachrichtentypen und dem HTTP-Header Content-Type zugeordnet.

HTTP POST

Welcher Nachrichtentyp am Ziel ankommt, hängt vom Wert des x-msg-class-Headers bzw. des Content-Type-Headers der HTTP-Anforderung ab.Tabelle 604 auf Seite 1276 zeigt den HTTP-Typ Content-Type , der jeder x-msg-classentspricht. Eines der Felder kann verwendet werden, um den Nachrichtentyp und das Nachrichtenformat festzulegen. Wenn beide Felder festgelegt und inkonsistent festgelegt sind, wird Bad Request exception zurückgegeben (HTTP 400, MQHTTP20004).

Tabelle 604. Zuordnung zwischen x-msg-class und HTTP Content-Type

x-msg-class	**HTTP Content-Type**
BYTES	application/octet-stream application/xml
TEXT	text/*
MAP	application/x-www-form-urlencoded application/xml (optional)
STREAM	application/xml (optional)

Wenn der JMS-Nachrichtentyp im MQRFH2-Header festgelegt wird, wird er gemäß Tabelle 605 auf Seite 1276 zugeordnet.

Tabelle 605. Zuordnung zwischen x-msg-class und JMS-Nachrichtentypen.

x-msg-class	**JMS-Nachrichtentyp**
BYTES	jms_bytes
TEXT	jms_text
MAP	jms_map
STREAM	jms_stream

Der JMS-Nachrichtentyp wird für die Nachrichtenklassen MAP und STREAM immer festgelegt. Für die Nachrichtenklassen BYTES und TEXT wird er nicht immer festgelegt. Wenn für die Anforderung ein MQRFH2-Header erstellt werden muss, wird der JMS-Nachrichtentyp immer festgelegt. Andernfalls, wenn kein MQRFH2-Header erstellt wird, wird kein JMS-Nachrichtentyp festgelegt. Ein MQRFH2-Header wird erstellt, wenn in der Anforderung Benutzereigenschaften mit dem x-msg-usr-Header festgelegt werden.

Wenn der JMS-Nachrichtentyp festgelegt wird, wird das Nachrichtenformat auf MQFMT_NONE festgelegt (siehe Tabelle 607 auf Seite 1277):

Tabelle 606. Zuordnung zwischen x-msg-class und dem WebSphere MQ-Nachrichtenformat

x-msg-class	**Nachrichtenformat, wenn MQRFH2-Header in Nachricht vorhanden**	**Nachrichtenformat, wenn *kein* MQRFH2-Header in Nachricht vorhanden**
BYTES	MQFMT_NONE	MQFMT_NONE

Tabelle 606. Zuordnung zwischen `x-msg-class` und dem WebSphere MQ-Nachrichtenformat (Forts.)		
`x-msg-class`	**Nachrichtenformat, wenn `MQRFH2`-Header in Nachricht vorhanden**	**Nachrichtenformat, wenn *kein* `MQRFH2`-Header in Nachricht vorhanden**
TEXT	MQFMT_NONE	MQFMT_STRING
MAP	MQFMT_NONE	Nicht möglich
STREAM	MQFMT_NONE	Nicht möglich

HTTP GET oder DELETE

Der abgerufene Nachrichtentyp bzw. das abgerufene Nachrichtenformat bestimmt den Wert des `x-msg-class`-Headers und des `Content-Type`-Headers der HTTP-Antwort. Der `x-msg-class`-Header wird nur zurückgegeben, wenn er in einer `x-msg-headers`-Anforderung angefordert wird.

In Tabelle 607 auf Seite 1277 werden die Zuordnungen zwischen `x-msg-class` und `Content-Type` sowie der aus der Warteschlange oder dem Thema abgerufene Nachrichtentyp beschrieben.

Tabelle 607. Nachrichtentypen zu `x-msg-class` und `Content-Type` zuordnen			
Nachrichtenformat	**JMS-Nachrichtentyp**	**`x-msg-class`**	**`Content-Type`**
Alles außer MQFMT_STRING	--	BYTES	application/octet-stream
MQFMT_STRING	--	TEXT	text/plain
MQFMT_NONE	jms_bytes	BYTES	application/octet-stream
MQFMT_NONE	jms_text	TEXT	text/plain
MQFMT_NONE	jms_map	MAP	application/xml
MQFMT_NONE	jms_stream	STREAM	application/xml

Serialisierung von MAP- und STREAM-Nachrichtenklassen

MAP- und STREAM-Nachrichtenklassen werden auf dieselbe Weise zum Client in der HTTP-Antwort zurück serialisiert, wie eine Nachricht in eine Warteschlange serialisiert wird.

Bei MAP werden die XML-Triplets aus Name, Typ und Wert wie folgt codiert:

```
<map>
    <elt name="elementname1" dt="datatype1">value1</elt>
    <elt name="elementname2" dt="datatype2">value2</elt>
    ...
</map>
```

STREAM ist ähnlich wie MAP, weist jedoch keine Elementnamen auf:

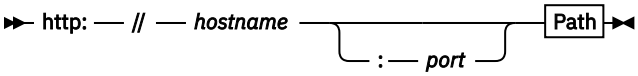
```
<stream>
    <elt dt="datatype1">value1</elt>
    <elt dt="datatype2">value2</elt>
    ...
</stream>
```

Anmerkung: `datatype` ist einer der Datentypen, die für die Definition von benutzerdefinierten Eigenschaften definiert und in „usr: HTTP-Entitätsheader x-msg-usr" auf Seite 1265 aufgeführt werden. Das Attribut `dt="string"` wird bei Zeichenfolgeelementen ausgelassen, da der Standarddatentyp `string` lautet.

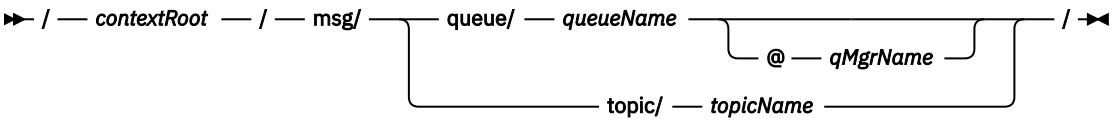
URI-Format

URIs, die von WebSphere MQ Bridge for HTTP abgefangen werden.

Syntax



Path



Anmerkung:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Beschreibung

Implementieren Sie das Servlet WebSphere MQ Bridge for HTTP auf Ihrem JEE-Anwendungsserver mit dem Kontextstammelement *contextRoot*. Anforderungen an

```
http://hostname:port/context_root/msg/queue/queueName@qMgrName
```

und

```
http://hostname:port/context_root/msg/topic/topicString
```

werden von WebSphere MQ Bridge for HTTP abgefangen.

IBM WebSphere MQ .NET-Klassen und -Schnittstellen

Die .NET-Klassen und -Schnittstellen von IBM WebSphere MQ sind alphabetisch aufgelistet. Die Eigenschaften, Methoden und Konstruktoren werden beschrieben.

.NET-Klasse MQAsyncStatus

Mit `MQAsyncStatus` können Sie den Status vorheriger MQI-Aktivitäten abfragen. Sie können beispielsweise den Erfolg einer vorherigen asynchronen PUT-Operation abfragen. `MQAsyncStatus` bindet die Funktionen der MQSTS- Datenstruktur ein.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```


- „Eigenschaften” auf Seite 1279
- „Konstruktoren” auf Seite 1279

Eigenschaften

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

`public static int CompCode {get;}`

Der Beendigungscode vom ersten Fehler oder von der ersten Warnung.

`public static int Reason {get;}`

Der Ursachencode vom ersten Fehler oder von der ersten Warnung.

`public static int PutSuccessCount {get;}`

Die Anzahl erfolgreicher asynchroner MQI-PUT-Aufrufe.

`public static int PutWarningCount {get;}`

Die Anzahl asynchroner MQI-PUT-Aufrufe, die erfolgreich mit einer Warnung abgeschlossen wurden.

`public static int PutFailureCount {get;}`

Die Anzahl fehlgeschlagener asynchroner MQI-PUT-Aufrufe.

`public static int ObjectType {get;}`

Der Objekttyp für den ersten Fehler. Folgende Werte sind möglich:

- `MQC.MQOT_ALIAS_Q`
- `MQC.MQOT_LOCAL_Q`
- `MQC.MQOT_MODEL_Q`
- `MQC.MQOT_Q`
- `MQC.MQOT_REMOTE_Q`
- `MQC.MQOT_TOPIC`
- `0` (es wird kein Objekt zurückgegeben)

`public static string ObjectName {get;}`

Der Objektname.

`public static string ObjectQMgrName {get;}`

Der Name des Objektwarteschlangenmanagers.

`public static string ResolvedObjectName {get;}`

Der aufgelöste Objektname.

`public static string ResolvedObjectQMgrName {get;}`

Der aufgelöste Name des Objektwarteschlangenmanagers.

Konstruktoren

`public MQAsyncStatus() throws MQException;`

Konstruktormethode, mit der ein Objekt mit Feldern erstellt wird, die anfangs nach Bedarf auf null gesetzt oder leer gelassen werden.

.NET-Klasse MQAuthenticationInformationRecord

Mit `MQAuthenticationInformationRecord` können Sie Informationen zu einem Authentifikator angeben, der in einer SSL-Clientverbindung in WebSphere MQ verwendet wird. `MQAuthenticationInformationRecord` bindet einen Authentifizierungsdatensatz, `MQAIR`, ein.

Klasse

```
System.Object
    └ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- „Eigenschaften” auf Seite 1280
- „Konstruktoren” auf Seite 1280

Eigenschaften

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

`public long Version {get; set;}`

Strukturversionsnummer.

`public long AuthInfoType {get; set;}`

Der Typ der Authentifizierungsdaten. Dieses Attribut muss auf einen der folgenden Werte gesetzt werden:

- OCSP - Die Überprüfung der Zertifikatswiderrufslisten erfolgt über OCSP.
- CRLLDAP - Die Überprüfung der Zertifikatswiderrufslisten erfolgt über Zertifikatswiderrufslisten auf LDAP-Servern.

`public string AuthInfoConnName {get; set;}`

Der DNS-Name oder die IP-Adresse des Hosts, auf dem der LDAP-Server ausgeführt wird, optional mit Angabe der Portnummer. Dieses Schlüsselwort ist erforderlich.

`public string LDAPPassword {get; set;}`

Das Kennwort, das dem definierten Namen des Benutzers, der auf den LDAP-Server zugreift, zugeordnet ist. Diese Eigenschaft gilt nur, wenn **AuthInfoType** auf CRLLDAP gesetzt ist.

`public string LDAPUserName {get; set;}`

Der definierte Name des Benutzers, der auf den LDAP-Server zugreift. Wenn Sie diese Eigenschaft festlegen, werden `LDAPUserNameLength` und `LDAPUserNamePtr` automatisch richtig festgelegt. Diese Eigenschaft gilt nur, wenn `AuthInfoType` auf CRLLDAP gesetzt ist.

`public string OCSPResponderURL {get; set;}`

Die URL, unter der der OCSP-Responder kontaktiert werden kann. Diese Eigenschaft gilt nur, wenn `AuthInfoType` auf OCSP gesetzt ist.

In diesem Feld wird die Groß-/Kleinschreibung beachtet. Der Eintrag muss mit der Zeichenfolge http:// in Kleinbuchstaben beginnen. Beim Rest der URL wird die Groß-/Kleinschreibung nur beachtet, wenn die OCSP-Serverimplementierung dies vorgibt.

Konstruktoren

`MQAuthenticationInformationRecord();`

.NET-Klasse `MQDestination`

Mit `MQDestination` können Sie auf Methoden zugreifen, die für `MQQueue` und `MQTopic` gleichermaßen gelten. `MQDestination` ist eine abstrakte Basisklasse und kann nicht instanziiert werden.

Klasse

```
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;
```


- „Eigenschaften” auf Seite 1281
- „Methoden” auf Seite 1281
- „Konstruktoren” auf Seite 1283

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public DateTime CreationDateTime {get;}

Datum und Uhrzeit der Erstellung dieser Warteschlange oder dieses Themas. Diese ursprünglich in MQQueue enthaltene Eigenschaft wurde in die Basisklasse MQDestination verschoben.

Es gibt keinen Standardwert.

public int DestinationType {get;}

Ganzzahliger Wert, der den Typ des verwendeten Ziels beschreibt. Vom Subklassenkonstruktor MQQueue oder MQTopic initialisiert, kann dieser Wert einen der folgenden Werte annehmen:

- MQOT_Q
- MQOT_TOPIC

Es gibt keinen Standardwert.

Methoden

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Löst MQException aus.

Ruft eine Nachricht aus einer Warteschlange ab, wenn das Ziel ein MQQueue-Objekt ist, oder aus einem Thema, wenn das Ziel ein MQTopic-Objekt ist. Der Abrufvorgang wird mit einer Standardinstanz von MQGetMessageOptions durchgeführt.

Wenn der Abrufvorgang fehlschlägt, bleibt das MQMessage-Objekt unverändert. Wenn der Vorgang erfolgreich ist, werden der Nachrichtendeskriptor und Teile der Nachrichtendaten von MQMessage durch den Nachrichtendeskriptor und die Nachrichtendaten aus der eingehenden Nachricht ersetzt.

Alle Aufrufe für WebSphere MQ von einem bestimmten MQQueueManager sind synchron. Wenn Sie daher einen Abruf mit Wartestatus durchführen, werden alle anderen Threads, die denselben MQQueueManager verwenden, blockiert und können keine weiteren WebSphere MQ-Aufrufe durchführen, bis der GET-Aufruf abgeschlossen ist. Wenn Sie mehrere Threads benötigen, um gleichzeitig auf WebSphere MQ zuzugreifen, muss jeder Thread sein eigenes MQQueueManager-Objekt erstellen.

message
Enthält der Nachrichtendeskriptor und die zurückgegebenen Nachrichtendaten. Einige Felder im Nachrichtendeskriptor sind Eingabeparameter. Es ist wichtig, sicherzustellen, dass die MessageId- und CorrelationId-Eingabeparameter als erforderlich festgelegt werden.

Ein Client, für den die Verbindung wiederhergestellt werden kann, gibt nach der erfolgreichen Verbindungswiederholung den Ursachencode MQRC_BACKED_OUT für Nachrichten zurück, die unter MQGM_SYNCPOINT empfangen wurden.

getMessageOptions
Optionen zum Steuern der Aktionen des Abrufs.

Die Verwendung von MQC.MQGMO_CONVERT führt möglicherweise zu einer Ausnahme mit dem Ursachencode MQC.MQRC_CONVERTED_STRING_TOO_BIG, wenn Sie eine Konvertierung von Codes mit Einzelbytezeichen in Codes mit Doppelbytezeichen durchführen. In diesem Fall wird die Nachricht ohne Konvertierung in den Zwischenspeicher kopiert.

Wenn *getMessageOptions* nicht angegeben wird, lautet die verwendete Nachrichtenoption MQGMO_NOWAIT.

Wenn Sie die Option MQGMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

MaxMsgSize
Die größte Nachricht, die dieses Nachrichtenobjekt erhalten darf. Wenn die Nachricht in der Warteschlange größer ist als dieser Wert, tritt eine der beiden folgenden Situationen ein:

- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag im MQGetMessageOptions-Objekt gesetzt ist, werden möglichst viele Nachrichtendaten in die Nachricht übernommen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_ACCEPTED ausgelöst.
- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag nicht gesetzt wird, wird die Nachricht in der Warteschlange belassen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_FAILED ausgelöst.

Wenn *MaxMsgSize* nicht angegeben ist, wird die gesamte Nachricht abgerufen.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Löst MQException aus.

Reiht eine Nachricht in eine Warteschlange ein, wenn das Ziel ein MQQueue-Objekt ist, oder veröffentlicht eine Nachricht in einem Thema, wenn das Ziel ein MQTopic-Objekt ist.

Änderungen am MQMessage-Objekt, die nach Abschluss des PUT-Aufrufs vorgenommen werden, wirken sich nicht auf die tatsächliche Nachricht in der WebSphere MQ-Warteschlange oder im Veröffentlichungsthema aus.

Mit Put werden die MessageId- und CorrelationId-Eigenschaften des MQMessage-Objekts aktualisiert. Dabei werden keine Nachrichtendaten gelöscht. Weitere Put- oder Get-Aufrufe beziehen sich auf die aktualisierten Informationen im MQMessage-Objekt. Beispiel: Im folgenden Codeausschnitt enthält die erste Nachricht a und die zweite ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Ein MQMessage-Objekt, das die Nachrichtendeskriptordaten und die zu sendende Nachricht enthält. Der Nachrichtendeskriptor kann sich aufgrund dieser Methode ändern. Die Werte im Nachrichtendeskriptor sofort nach Abschluss dieser Methode sind die Werte, die in die Warteschlange eingereiht oder für das Thema veröffentlicht wurden.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- MQRC_CALL_INTERRUPTED, wenn die Verbindung unterbrochen wird, während ein PUT-Aufruf für eine persistente Nachricht durchgeführt wird und die Verbindungswiederholung erfolgreich ist.
- MQRC_NONE, wenn die Verbindung erfolgreich hergestellt wurde, während ein PUT-Aufruf für eine nicht persistente Nachricht durchgeführt wurde (siehe Anwendungswiederherstellung).

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessage-Optons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

Anmerkung: Wenn Sie aus Gründen der Einfachheit und Leistungsoptimierung eine einzelne Nachricht in eine Warteschlange einreihen möchten, verwenden Sie das MQQueueManager.Put-Objekt. Hierfür sollten Sie über ein MQQueue-Objekt verfügen.

Konstruktoren
MQDestination ist eine abstrakte Basisklasse und kann nicht instanziiert werden. Greifen Sie mit MQQueue -und MQTopic -Konstruktoren oder mit MQQueueManager.AccessQueue und MQQueueManager.AccessTopic methodsauf Ziele zu.

.NET-Klasse MQEnvironment
Mit MQEnvironment können Sie steuern, wie der MQQueueManager-Konstruktor genannt wird. Außerdem können Sie damit eine WebSphere MQ-MQI-Clientverbindung auswählen. Die Klasse MQEnvironment enthält Eigenschaften zum Steuern des Verhaltens von WebSphere MQ.

Klasse

```
System.Object
    │
    └─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```


- „Eigenschaften - nur Client” auf Seite 1283
- „Eigenschaften” auf Seite 1284
- „Konstruktoren” auf Seite 1285

Eigenschaften - nur Client
Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public static int CertificateValPolicy {get; set;}

Legen Sie fest, welche SSL/TLS-Zertifikatprüfrichtlinie verwendet wird, um digitale Zertifikate, die von fernen Partnersystemen empfangen werden, auf Gültigkeit zu prüfen. Gültige Werte sind:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

public static ArrayList EncryptionPolicySuiteB {get; set;}

Legen Sie die Ebene der mit Suite B kompatiblen Verschlüsselung fest. Gültige Werte sind:

- MQC.MQ_SUITE_B_NONE - Dies ist der Standardwert.

- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}
Der Name des Kanals für die Verbindung zum Zielwarteschlangenmanager. Sie *müssen* die Kanaleigenschaft festlegen, bevor Sie eine Instanz von `MQQueueManager` im Clientmodus instanziieren.

public static int FipsRequired {get; set;}

Geben Sie `MQC.MQSSL_FIPS_YES` an, damit für Verschlüsselungen in WebSphere MQ nur FIPS-zertifizierte Algorithmen verwendet werden. Die Standardeinstellung ist `MQC.MQSSL_FIPS_NO`.

Wenn Verschlüsselungshardware konfiguriert ist, werden die Verschlüsselungsmodule verwendet, die vom Hardwareprodukt bereitgestellt werden. Abhängig von der verwendeten Hardware sind diese Module möglicherweise bis zu einer bestimmten Ebene nicht FIPS-zertifiziert.

public static string Hostname {get; set;}
Der TCP/IP-Hostname des Computers, auf dem sich der WebSphere MQ-Server befindet. Wenn der Hostname nicht festgelegt wird und keine überschreibenden Eigenschaften festgelegt wurden, wird für die Verbindung zum lokalen Warteschlangenmanager der Serverbindungsmodus verwendet.

public static int Port {get; set;}
Der Port für die Verbindung. An diesem Port ist der WebSphere MQ-Server empfangsbereit für eingehende Verbindungsanforderungen. Der Standardwert ist 1414.

public static string SSLCipherSpec {get; set;}
Setzen Sie `SSLCipherSpec` auf den Wert für CipherSpec, der im SVRCONN-Kanal festgelegt ist, um SSL für die Verbindung zu aktivieren. Der Standardwert ist null, und SSL ist für die Verbindung nicht aktiviert.

public static string sslPeerName {get; set;}

Ein Muster für einen definierten Namen. Wenn `sslCipherSpec` festgelegt wurde, können Sie mit dieser Variablen sicherstellen, dass der richtige Warteschlangenmanager verwendet wird. Wenn dieser Wert auf null (Standardeinstellung) gesetzt wird, wird der definierte Name des Warteschlangenmanagers nicht ausgeführt. `sslPeerName` wird ignoriert, wenn `sslCipherSpec` null ist.

Eigenschaften

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

public static ArrayList HdrCompList {get; set;}
Liste für die Komprimierung der Headerdaten

public static int KeyResetCount {get; set;}
Die Anzahl an unverschlüsselten Bytes, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet bzw. empfangen werden.

public static ArrayList MQAIRArray {get; set;}
Ein Bereich mit `MQAuthenticationInformationRecord`-Objekten.

public static ArrayList MsgCompList {get; set;}
Liste für die Nachrichtendatenkomprimierung

public static string Password {get; set;}
Das zu authentifizierende Kennwort. Das Kennwort, auf das die `MQCSP`-Struktur verweist, wird automatisch eingetragen, wenn Sie diese Kennworteigenschaft festlegen.

public static string ReceiveExit {get; set;}
Mit einem Empfangsexit können Sie von einem Warteschlangenmanager empfangene Daten prüfen und ändern. Er wird in der Regel zusammen mit einem entsprechenden Sendeexit beim Warteschlangenmanager verwendet. Wenn "ReceiveExit" auf null gesetzt wird, wird kein Empfangsexit abgerufen.

public static string ReceiveUserData {get; set;}
Die einem Empfangsexit zugeordneten Benutzerdaten. Der Wert ist auf 32 Zeichen begrenzt.

public static string SecurityExit {get; set;}
Mit einem Sicherheitsexit können Sie die Sicherheitsnachrichtenflüsse anpassen, die bei dem Versuch auftreten, eine Verbindung zu einem Warteschlangenmanager herzustellen. Wenn `SecurityExit` auf null gesetzt wird, wird kein Sicherheitsexit abgerufen.

public static string SecurityUserData {get; set;}
Die einem Sicherheitsexit zugeordneten Benutzerdaten. Der Wert ist auf 32 Zeichen begrenzt.

public static string SendExit {get; set;}
Mit einem Sendeexit können Sie an einen Warteschlangenmanager gesendete Daten prüfen und ändern. Er wird in der Regel zusammen mit einem entsprechenden Empfangsexit beim Warteschlangenmanager verwendet. Wenn `SendExit` auf null gesetzt wird, wird kein Sendeexit abgerufen.

public static string SendUserData {get; set;}
Die einem Sendeexit zugeordneten Benutzerdaten. Der Wert ist auf 32 Zeichen begrenzt.

public static string SharingConversations {get; set;}

Das Feld `SharingConversations` wird für Verbindungen von .NET-Anwendungen verwendet, wenn diese Anwendungen keine Definitionstabelle für Clientkanäle (CCDT) verwenden.

Mit `SharingConversations` wird die maximale Anzahl Dialoge festgelegt, die an einem dieser Verbindung zugeordneten Socket gemeinsam genutzt werden können.

Der Wert 0 bedeutet, dass der Kanal hinsichtlich der gemeinsamen Nutzung von Dialogen, dem Vorauslesen und den Überwachungssignalen genauso funktioniert wie vor WebSphere MQ Version 7.0.

Das Feld wird in der Hashtabelle der Eigenschaften als `SHARING_CONVERSATIONS_PROPERTY` übergeben, wenn Sie einen WebSphere MQ-Warteschlangenmanager instanziieren.

Wenn Sie `SharingConversations` nicht angeben, wird der Standardwert 10 verwendet.

public static string SSLCryptoHardware {get; set;}

Legt den Namen der Parameterzeichenfolge fest, die für die Konfiguration der Verschlüsselungshardware auf dem System erforderlich ist. `SSLCryptoHardware` wird ignoriert, wenn `sslCipherSpec` null ist.

public static string SSLKeyRepository {get; set;}

Legen Sie den vollständig qualifizierten Dateinamen des Schlüsselrepositorys an.

Wenn `SSLKeyRepository` auf null gesetzt wird (Standardeinstellung), wird die Umgebungsvariable `MQSSLKEYR` des Zertifikats verwendet, um das Schlüsselrepository zu suchen. `SSLCryptoHardware` wird ignoriert, wenn `sslCipherSpec` null ist.

Anmerkung: Die Erweiterung `.kdb` ist ein obligatorischer Bestandteil des Dateinamens, ist jedoch nicht als Teil des Parameterwerts eingeschlossen. Das Verzeichnis, das Sie angeben, muss bereits vorhanden sein. WebSphere MQ erstellt die Datei beim ersten Zugriff auf das neue Schlüsselrepository, falls die Datei noch nicht vorhanden ist.

public static string UserId {get; set;}

Die zu authentifizierende Benutzer-ID. Die Benutzer-ID, auf die die `MQCSP`-Struktur verweist, wird automatisch eingetragen, wenn Sie `UserId` festlegen. Authentifizieren Sie `UserId` anhand einer API oder eines Sicherheitsexits.

Konstruktoren

public MQEnvironment()

.NET-Klasse `MQException`

Mit `MQException` können Sie den Beendigungs- und den Ursachencode einer fehlgeschlagenen WebSphere MQ-Funktion feststellen. `MQException` wird ausgelöst, wenn ein WebSphere MQ-Fehler auftritt.

Klasse

```
System.Object
    └ System.Exception
        └ System.ApplicationException
            └ IBM.WMQ.MQException
```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```

- „Eigenschaften” auf Seite 1286
- „Konstruktoren” auf Seite 1286

Eigenschaften

public int CompletionCode {get; set;}
Der dem Fehler zugeordnete WebSphere MQ-Beendigungscode. Folgende Werte sind möglich:

- `MQException.MQCC_OK`
- `MQException.MQCC_WARNING`
- `MQException.MQCC_FAILED`

public int ReasonCode {get; set;}
Der WebSphere MQ-Ursachencode, der den Fehler beschreibt.

Konstruktoren

public MQException(int *completionCode*, int *reasonCode*)

completionCode
Der WebSphere MQ-Beendigungscode.

reasonCode
Der WebSphere MQ-Beendigungscode.

.NET-Klasse MQGetMessageOptions

Mit `MQGetMessageOptions` können Sie angeben, wie Nachrichten abgerufen werden sollen. Mit dieser Klasse wird das Verhalten von `MQDestination.Get` geändert.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- „Eigenschaften” auf Seite 1286
- „Konstruktoren” auf Seite 1289

Eigenschaften

Anmerkung: Das Verhalten einiger der in dieser Klasse verfügbaren Optionen hängt von der Umgebung ab, in der sie verwendet werden. Diese Elemente sind mit einem Stern (*) markiert.

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

public int GroupStatus {get;}*

Mit `GroupStatus` wird angegeben, ob sich die abgerufene Nachricht in einer Gruppe befindet und ob sie das letzte Element in der Gruppe ist. Mögliche Werte:

MQC.MQGS_LAST_MSG_IN_GROUP
Nachricht ist die letzte oder einzige Nachricht in der Gruppe.

MQC.MQGS_MSG_IN_GROUP
Nachricht befindet sich in einer Gruppe, ist jedoch nicht die letzte in der Gruppe.

MQC.MQGS_NOT_IN_GROUP
Nachricht befindet sich nicht in einer Gruppe.

public int MatchOptions {get; set;}*

Mit `MatchOptions` wird festgelegt, wie eine Nachricht ausgewählt wird. Sie können folgende Optionen für den Abgleich festlegen:

MQC.MQMO_MATCH_CORREL_ID
Die Korrelations-ID, mit der ein Abgleich durchgeführt wird.

MQC.MQMO_MATCH_GROUP_ID
Die Gruppen-ID, mit der ein Abgleich durchgeführt wird.

MQC.MQMO_MATCH_MSG_ID
Die Nachrichten-ID, mit der ein Abgleich durchgeführt wird.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER
Die Nachrichtenfolgenummer soll abgeglichen werden.

MQC.MQMO_NONE
Kein Abgleich erforderlich.

public int Options {get; set;}

Mit `Options` wird die Aktion von `MQQueue.get` gesteuert. Sie können einen beliebigen der folgenden Werte angeben. Wenn mehrere Optionen erforderlich sind, können die Werte hinzugefügt oder mithilfe des bitweisen ODER-Operators kombiniert werden.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG
Abschneiden der Nachrichtendaten zulassen

MQC.MQGMO_ALL_MSGS_AVAILABLE*
Nachrichten nur dann aus einer Gruppe abrufen, wenn alle Nachrichten in der Gruppe verfügbar sind.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*
Die Segmente einer logischen Nachricht nur dann abrufen, wenn alle Segmente in der Gruppe verfügbar sind.

MQC.MQGMO_BROWSE_FIRST
Anfang der Warteschlange anzeigen

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*
Nachricht unter Anzeigecursor anzeigen

MQC.MQGMO_BROWSE_NEXT
Ab der aktuellen Position in der Warteschlange durchsuchen.

MQC.MQGMO_COMPLETE_MSG*
Nur vollständige logische Nachrichten abrufen.

MQC.MQGMO_CONVERT
Hiermit können Sie die zu konvertierenden Anwendungsdaten anfordern, damit diese den Attributen `CharacterSet` und `Encoding` von `MQMessage` entsprechen, bevor die Daten in den Nachrichtenpuffer kopiert werden. Da die Datenkonvertierung auch ausgeführt wird, wenn die Daten aus dem Nachrichtenpuffer abgerufen werden, legen Anwendungen diese Option nicht fest.

Wenn Sie diese Option verwenden, können beim Konvertieren von Einzelbytezeichensätzen in Doppelbytezeichensätze Probleme auftreten. Nehmen Sie die Konvertierung stattdessen mit den Methoden `readString`, `readLine` und `writeString` vor, nachdem die Nachricht zugestellt wurde.

MQC.MQGMO_FAIL_IF_QUIESCING
Fehler, wenn der Warteschlangenmanager stillgelegt wird.

MQC.MQGMO_LOCK*
Durchsuchte Nachricht sperren.

MQC.MQGMO_LOGICAL_ORDER*
Nachrichten in Gruppen und Segmenten logischer Nachrichten in logischer Reihenfolge zurückgeben.

Wenn Sie die Option MQGMO_LOGICAL_ORDER bei einem Client verwenden, für den die Verbindung wiederhergestellt werden kann, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE an die Anwendung zurückgegeben.

MQC.MQGMO_MARK_SKIP_BACKOUT*
Eine Arbeitseinheit darf zurückgesetzt werden, ohne die Nachricht in der Warteschlange wiederherzustellen.

MQC.MQGMO_MSG_UNDER_CURSOR
Nachricht unter Anzeigecursor abrufen

MQC.MQGMO_NONE
Es wurden keine anderen Optionen angegeben; alle Optionen nehmen ihre Standardwerte an.

MQC.MQGMO_NO_PROPERTIES
Es werden keine Eigenschaften der Nachricht, mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung), abgerufen.

MQC.MQGMO_NO_SYNCPOINT
Nachricht ohne Synchronisationspunktsteuerung abrufen.

MQC.MQGMO_NO_WAIT
Sofort zurückgeben, wenn keine geeignete Nachricht vorliegt.

MQC.MQGMO_PROPERTIES_AS_Q_DEF
Nachrichteneigenschaften abrufen, wie im Attribut PropertyControl von MQQueue definiert. Der Zugriff auf die Nachrichteneigenschaften im Nachrichtendeskriptor bzw. in der Erweiterung ist vom Attribut PropertyControl nicht betroffen.

MQC.MQGMO_PROPERTIES_COMPATIBILITY
Nachrichteneigenschaften mit einem Präfix mcd, jms, usr oder mqext in MQRFH2-Headern abrufen. Andere Eigenschaften der Nachricht, mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung), werden verworfen.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2
Nachrichteneigenschaften, mit Ausnahme der Eigenschaften im Nachrichtendeskriptor (bzw. in der Erweiterung), in MQRFH2-Headern abrufen. Verwenden Sie MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 in Anwendungen, die das Abrufen von Eigenschaften erwarten, jedoch nicht so geändert werden können, dass sie Nachrichtenkennungen verwenden können.

MQC.MQGMO_PROPERTIES_IN_HANDLE
Nachrichteneigenschaften mithilfe von MsgHandle abrufen.

MQC.MQGMO_SYNCPOINT
Nachricht mit Synchronisationspunktsteuerung abrufen. Die Nachricht ist als nicht verfügbar für andere Anwendungen markiert, wird jedoch nur dann aus der Warteschlange gelöscht, wenn die Arbeitseinheit festgeschrieben wird. Die Nachricht steht wieder zur Verfügung, wenn die Arbeitseinheit zurückgesetzt wird.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*
Nachricht mit Synchronisationspunktsteuerung abrufen, wenn die Nachricht persistent ist.

MQC.MQGMO_UNLOCK*
Zuvor gesperrte Nachricht entsperren.

MQC.MQGMO_WAIT
Auf den Eingang einer Nachricht warten.

public string ResolvedQueueName {get;}
Der Warteschlangenmanager setzt `ResolvedQueueName` auf den lokalen Namen der Warteschlange, aus der die Nachricht abgerufen wurde. `ResolvedQueueName` unterscheidet sich von dem Namen, der zum Öffnen der Warteschlange verwendet wurde, als eine Aliaswarteschlange oder Modellwarteschlange geöffnet wurde.

public char Segmentation {get;}*
Mit `Segmentation` wird angegeben, ob die Segmentierung für die abgerufene Nachricht zulässig ist. Mögliche Werte:

MQC.MQSEG_INHIBITED
Keine Segmentierung zulassen.

MQC.MQSEG_ALLOWED
Segmentierung zulassen.

public byte SegmentStatus {get;}*
Bei `SegmentStatus` handelt es sich um ein Ausgabefeld, mit dem angegeben wird, ob die abgerufene Nachricht ein Segment einer logischen Nachricht ist. Wenn die Nachricht ein Segment ist, wird mit dem Flag angegeben, ob es sich um das letzte Segment handelt. Mögliche Werte:

MQC.MQSS_LAST_SEGMENT
Nachricht ist das letzte oder einzige Segment der logischen Nachricht.

MQC.MQSS_NOT_A_SEGMENT
Nachricht ist kein Segment.

MQC.MQSS_SEGMENT
Nachricht ist ein Segment, aber nicht das letzte Segment der logischen Nachricht

public int WaitInterval {get; set;}
`WaitInterval` ist der maximale Zeitraum in Millisekunden, den ein `MQQueue.get`-Aufruf auf den Eingang einer geeigneten Nachricht wartet. Verwenden Sie `WaitInterval` mit `MQC.MQGMO_WAIT`. Legen Sie den Wert `MQC.MQWI_UNLIMITED` fest, um einen unbegrenzten Zeitraum auf eine Nachricht zu warten.

Konstruktoren

public MQGetMessageOptions()
Erstellen Sie ein neues `MQGetMessageOptions`-Objekt, bei dem `Options` auf `MQC.MQGMO_NO_WAIT`, `WaitInterval` auf null und `ResolvedQueueName` auf einen leeren Wert gesetzt ist.

.NET-Klasse MQManagedObject

Mit `MQManagedObject` können Sie Attribute für `MQDestination`, `MQProcess`, `MQQueueManager` und `MQSubscription` abfragen und festlegen. `MQManagedObject` ist eine Superklasse dieser Klassen.

Klassen

```
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;
```


Eigenschaften

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

public string AlternateUserId {get; set;}
Die alternative Benutzer-ID, die gegebenenfalls beim Öffnen der Ressource festgelegt wurde. `AlternateUserID.set` wird ignoriert, wenn die Option für ein geöffnetes Objekt ausgegeben wird. `AlternateUserId` ist für Subskriptionen ungültig.

public int CloseOptions {get; set;}

Legen Sie dieses Attribut fest, um zu steuern, wie die Ressource geschlossen wird. Der Standardwert ist `MQC.MQCO_NONE`. `MQC.MQCO_NONE` ist der einzige zulässige Wert für alle anderen Ressourcen als permanente dynamische Warteschlangen, temporäre dynamische Warteschlangen, Subskriptionen und Themen, auf die von den Objekten zugegriffen wird, die sie erstellt haben.

Für Warteschlangen und Themen sind folgende zusätzliche Werte gültig:

MQC.MQCO_DELETE
Warteschlange löschen, wenn keine Nachrichten vorliegen.

MQC.MQCO_DELETE_PURGE
Die Warteschlange zusammen mit allen in ihr enthaltenen Nachrichten löschen.

MQC.MQCO_QUIESCE
Anfordern, dass die Warteschlange geschlossen wird und Sie eine Warnung erhalten, wenn noch Nachrichten vorhanden sind (diese können vor dem endgültigen Schließen abgerufen werden).

Für Subskriptionen sind folgende zusätzliche Werte gültig:

MQC.MQCO_KEEP_SUB
Die Subskription wird nicht gelöscht. Diese Option ist nur gültig, wenn die ursprüngliche Subskription permanent ist. `MQC.MQCO_KEEP_SUB` ist der Standardwert für ein permanentes Thema.

MQC.MQCO_REMOVE_SUB
Die Subskription wird gelöscht. `MQC.MQCO_REMOVE_SUB` ist der Standardwert für ein nicht permanentes, nicht verwaltetes Thema.

MQC.MQCO_PURGE_SUB
Die Subskription wird gelöscht. `MQC.MQCO_PURGE_SUB` ist der Standardwert für ein nicht permanentes, verwaltetes Thema.

public MQQueueManager ConnectionReference {get;}
Der Warteschlangenmanager, zu dem diese Ressource gehört.

public string MQDescription {get;}
Die Beschreibung der Ressource, wie sie auf dem Warteschlangenmanager gespeichert ist. `MQDescription` gibt eine leere Zeichenfolge für Subskriptionen und Themen zurück.

public boolean IsOpen {get;}
Gibt an, ob die Ressource derzeit geöffnet ist.

public string Name {get;}
Der Name der Ressource. Der Name wird entweder bei der Zugriffsmethode bereitgestellt oder vom Warteschlangenmanager für eine dynamische Warteschlange zugeordnet.

public int OpenOptions {get; set;}
"OpenOptions" werden festgelegt, wenn ein WebSphere MQ-Objekt geöffnet wird. Die Methode `OpenOptions.set` wird ignoriert und löst keinen Fehler aus. Abonnements haben keine `OpenOptions`.

Methoden

public virtual void Close();

Löst MQException aus.

Schließt das Objekt. Für diese Ressource sind nach dem Aufrufen von Close keine weiteren Vorgänge mehr zulässig. Um das Verhalten der Methode Close zu ändern, legen Sie das Attribut closeOptions fest.

public string GetAttributeString(int *selector*, int *length*);

Löst MQException aus.

Ruft eine Attributzeichenfolge ab.

selector
Ganzzahl zur Angabe, welches Attribut abgefragt wird.

length
Ganzzahl zur Angabe der Länge der erforderlichen Zeichenfolge.

public void Inquire(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);

Löst MQException aus.

Gibt eine Ganzzahlenfeldgruppe und einen Satz Zeichenfolgen zurück, die die Attribute einer Warteschlange, eines Prozesses oder des Warteschlangenmanagers enthalten. Die abzufragenden Attribute werden in der Selektorenfeldgruppe angegeben.

Anmerkung: Viele der gebräuchlicheren Attribute können mit den Get-Methoden abgefragt werden, die in MQManagedObject, MQQueue und MQQueueManager definiert sind.

selectors
Ganzzahlenfeldgruppe zur Angabe der Attribute mit Werten, die abgefragt werden sollen.

intAttrs
Die Feldgruppe, in die die Ganzzahlenattributwerte zurückgegeben werden. Ganzzahlenattributwerte werden in derselben Reihenfolge wie die Ganzzahlenattributselektoren in der Selektorenfeldgruppe zurückgegeben.

charAttrs
Der Puffer, in dem die Zeichenattribute in verketteter Form zurückgegeben werden. Zeichenattribute werden in derselben Reihenfolge wie die Zeichenattributselektoren in der Selektorenfeldgruppe zurückgegeben. Die Länge jeder Attributzeichenfolge ist für jedes Attribut festgelegt.

public void Set(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);

Löst MQException aus.

Legt die im Vektor der Selektoren definierten Attribute fest. Die festzulegenden Attribute werden in der Selektorenfeldgruppe angegeben.

selectors
Ganzzahlenfeldgruppe zur Angabe der Attribute mit abzufragenden Werten.

intAttrs
Die Feldgruppe der Ganzzahlenattributwerte, die festgelegt werden sollen. Diese Werte müssen dieselbe Reihenfolge aufweisen wie die Ganzzahlenattributselektoren in der Selektorenfeldgruppe.

charAttrs
Der Puffer, in dem die festzulegenden Zeichenattribute verkettet werden. Diese Werte müssen dieselbe Reihenfolge aufweisen wie die Zeichenattributselektoren in der Selektorenfeldgruppe. Die Länge jedes Zeichenattributs ist festgelegt.

public void SetAttributeString(int *selector*, string *value*, int *length*);

Löst MQException aus.

Legt eine Attributzeichenfolge fest.

selector
Ganzzahl zur Angabe, welches Attribut festgelegt wird.

value
Die Zeichenfolge, die als Attributwert festgelegt werden soll.

length
Ganzzahl zur Angabe der Länge der erforderlichen Zeichenfolge.

Konstruktoren

protected MQManagedObject()
Konstruktormethode. Bei diesem Objekt handelt es sich um eine abstrakte Basisklasse, die nicht von sich selbst instanziiert werden kann.

.NET-Klasse MQMessage

Mit MQMessage können Sie auf den Nachrichtendeskriptor und die Daten für eine WebSphere MQ-Nachricht zugreifen. MQMessage bindet eine WebSphere MQ-Nachricht ein.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Erstellen Sie ein MQMessage-Objekt und verwenden Sie anschließend die Methoden Read und Write, um Daten zwischen der Nachricht und anderen Objekten in Ihrer Anwendung zu übertragen. Senden und empfangen Sie MQMessage-Objekte mit den Methoden Put und Get der Klassen MQDestination, MQQueue und MQTopic.

Verwenden Sie zum Abrufen und Festlegen der Eigenschaften des Nachrichtendeskriptors die Eigenschaften von MQMessage. Verwenden Sie zum Festlegen und Abrufen erweiterter Nachrichteneigenschaften die Methoden SetProperty und GetProperty.

- „Eigenschaften” auf Seite 1292
- „Die Nachrichtenmethoden Read und Write” auf Seite 1298
- „Puffermethoden” auf Seite 1301
- „Eigenschaftenmethoden” auf Seite 1301
- „Konstruktoren” auf Seite 1303

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public string AccountingToken {get; set;}
Teil des Identitätskontexts einer Nachricht; unterstützt eine Anwendung dabei, den Aufwand für die als Ergebnis einer Nachricht durchgeführte Arbeit zu bestimmen. Der Standardwert ist MQC.MQACT_NONE.

public string ApplicationIdData {get; set;}
Teil des Identitätskontexts einer Nachricht. ApplicationIdData enthält Informationen, die in der Anwendungssuite definiert wurden, und kann zusätzliche Informationen über die Nachricht oder den Absender zur Verfügung stellen. Der Standardwert ist " ".

public string ApplicationOriginData {get; set;}
Von der Anwendung definierte Informationen, die zur Bereitstellung zusätzlicher Informationen über den Ursprung der Nachricht verwendet werden können. Der Standardwert ist " ".

public int BackoutCount {get;}
Ein Zähler dafür, wie häufig die Nachricht zuvor von einem MQQueue.Get-Aufruf als Teil einer Arbeitseinheit zurückgesendet und zurückgesetzt wurde. Der Standardwert ist null.

public int CharacterSet {get; set;}

Die ID des codierten Zeichensatzes für die Zeichendaten in der Nachricht.

Legen Sie CharacterSet fest, um den Zeichensatz für die Zeichendaten in der Nachricht anzugeben. Rufen Sie CharacterSet ab, um festzustellen, welcher Zeichensatz zur Codierung der Zeichendaten in der Nachricht verwendet wurde.

.NET-Anwendungen werden immer in Unicode ausgeführt, während Anwendungen in anderen Umgebungen mit demselben Zeichensatz ausgeführt werden, mit dem sie im Warteschlangenmanager ausgeführt werden.

Mit den Methoden ReadString und ReadLine werden die Zeichendaten in der Nachricht in das Unicode-Format konvertiert.

Mit der Methode WriteString wird die Konvertierung aus dem Unicode-Format in den unter CharacterSet codierten Zeichensatz durchgeführt. Wenn CharacterSet auf den entsprechenden Standardwert, MQC.MQCCSI_Q_MGR ungleich 0, gesetzt wird, erfolgt keine Konvertierung, und CharacterSet wird auf 1200 gesetzt. Wenn Sie CharacterSet auf einen anderen Wert setzen, führt WriteString die Konvertierung aus dem Unicode-Format in den alternativen Wert durch.

Anmerkung: Andere Lese- und Schreibmethoden verwenden CharacterSet nicht.

- Mit ReadChar und WriteChar wird ein Unicode-Zeichen ohne Konvertierung aus einem Nachrichtenpuffer gelesen oder in einen Nachrichtenpuffer geschrieben.
- Mit ReadUTF und WriteUTF wird eine Konvertierung zwischen einer Unicode-Zeichenfolge in der Anwendung und einer UTF-8-Zeichenfolge mit vorangestelltem, aus 2 Bytes bestehendem Feld im Nachrichtenpuffer durchgeführt.
- Bytemethoden übertragen Bytes unverändert zwischen der Anwendung und dem Nachrichtenpuffer.

public byte[] CorrelationId {get; set;}

- Bei einem MQQueue.Get-Aufruf ist dies die Korrelations-ID der abzurufenden Nachricht. Der Warteschlangenmanager gibt die erste Nachricht mit einer Nachrichten-ID und einer Korrelations-ID zurück, die mit den Feldern des Nachrichtendeskriptors übereinstimmen. Der Standardwert MQC.MQCI_NONE sorgt dafür, dass alle Korrelations-IDs übereinstimmen.
- Bei einem MQQueue.Put-Aufruf ist dies die festzulegende Korrelations-ID.

public int DataLength {get;}
Die Anzahl der Bytes der verbleibenden Nachrichtendaten, die noch gelesen werden müssen.

public int DataOffset {get; set;}
Die aktuelle Cursorposition in den Nachrichtendaten. Lese- und Schreibvorgänge finden an der aktuellen Position statt.

public int Encoding {get; set;}
Die für numerische Werte in den Anwendungsnachrichtendaten verwendete Darstellung. Encoding gilt für binäre, gepackte Dezimalzahlen und Gleitkommadaten. Das Verhalten der Lese- und Schreibmethoden für diese numerischen Formate wird entsprechend geändert. Erstellen Sie einen Wert für das Codierungsfeld, indem Sie einen Wert aus einem dieser drei Abschnitte hinzufügen. Alternativ dazu können Sie den Wert erstellen, indem Sie die Werte aus allen drei Abschnitten mithilfe des bitweisen ODER-Operators kombinieren.

1. Binäre Ganzzahl

   MQC.MQENC_INTEGER_NORMAL
   Big-Endian-Ganzzahlen.

   MQC.MQENC_INTEGER_REVERSED
   Little-Endian-Ganzzahlen, wie sie in der Intel-Architektur verwendet werden.

2. Gepackte Dezimalzahl

MQC.MQENC_DECIMAL_NORMAL
Gepackte Big-Endian-Dezimalzahl, wie sie in z/OS verwendet wird.

MQC.MQENC_DECIMAL_REVERSED
Gepackte Little-Endian-Dezimalzahl.

3. Gleitkomma

MQC.MQENC_FLOAT_IEEE_NORMAL
Big-Endian-IEEE-Gleitkommazahlen.

MQC.MQENC_FLOAT_IEEE_REVERSED
Little-Endian-IEEE-Gleitkommazahlen, wie sie in der Intel-Architektur verwendet werden.

MQC.MQENC_FLOAT_S390
Gleitkommazahlen im z/OS-Format.

Der Standardwert ist:

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

Bei der Standardeinstellung schreibt `WriteInt` eine Little-Endian-Ganzzahl, und `ReadInt` liest eine Little-Endian-Ganzzahl. Wenn Sie stattdessen das Flag `MQC.MQENC_INTEGER_NORMAL` setzen, schreibt `WriteInt` eine Big-Endian-Ganzzahl und `ReadInt` liest eine Big-Endian-Ganzzahl.

Anmerkung: Beim Konvertieren von Gleitkommazahlen im IEEE-Format in Gleitkommazahlen im zSeries-Format können Fehler in der Genauigkeit auftreten.

public int Expiry {get; set;}
Eine Ablaufzeit, die in Zehntel Sekunden ausgedrückt und von der Anwendung festgelegt wird, die die Nachricht einreiht. Nachdem die Ablaufzeit einer Nachricht verstrichen ist, kann die Nachricht vom Warteschlangenmanager verworfen werden. Wenn die Nachricht eines der `MQC.MQRO_EXPIRATION`-Flags angegeben hat, wird ein Bericht erstellt, wenn die Nachricht verworfen wird. Der Standardwert ist `MQC.MQEI_UNLIMITED`, d. h. dass die Nachricht nie abläuft.

public int Feedback {get; set;}
Verwenden Sie `Feedback` mit einer Nachricht des Typs `MQC.MQMT_REPORT`, um die Art des Berichts anzugeben. Folgende Rückmeldungscodes werden vom System definiert:

- `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`

Anwendungsdefinierte Rückmeldungswerte zwischen `MQC.MQFB_APPL_FIRST` und `MQC.MQFB_APPL_LAST` können ebenfalls verwendet werden. Der Standardwert für dieses Feld ist `MQC.MQFB_NONE`. Damit wird angegeben, dass keine Rückmeldung bereitgestellt wird.

public string Format {get; set;}
Ein Formatname, mit dem der Sender der Nachricht dem Empfänger zu erkennen gibt, welche Art von Daten die Nachricht enthält. Sie können eigene Formatnamen verwenden, müssen dabei aber beach-

ten, dass Namen, die mit den Buchstaben MQ beginnen, bereits über vom Warteschlangenmanager definierte Bedeutungen verfügen. Folgende Formate sind im Warteschlangenmanager integriert:

MQC.MQFMT_ADMIN
Anforderungs-/Antwortnachricht des Befehlsservers

MQC.MQFMT_COMMAND_1
Antwortnachricht für Befehle vom Typ 1

MQC.MQFMT_COMMAND_2
Antwortnachricht für Befehle vom Typ 2

MQC.MQFMT_DEAD_LETTER_HEADER
Header für nicht zustellbare Nachrichten

MQC.MQFMT_EVENT
Ereignisnachricht

MQC.MQFMT_NONE
Kein Formatname

MQC.MQFMT_PCF
Benutzerdefinierte Nachricht im Programmable Command Format.

MQC.MQFMT_STRING
Nachricht, die nur aus Zeichen besteht

MQC.MQFMT_TRIGGER
Auslösenachricht

MQC.MQFMT_XMIT_Q_HEADER
Header der Übertragungswarteschlange

Der Standardwert ist MQC.MQFMT_NONE.

public byte[] GroupId {get; set;}
Eine Bytefolge zur Angabe der Nachrichtengruppe, zu der die physische Nachricht gehört. Der Standardwert ist MQC.MQGI_NONE.

public int MessageFlags {get; set;}
Flags zum Steuern der Segmentierung und des Status einer Nachricht.

public byte[] MessageId {get; set;}

Bei einem MQQueue.Get-Aufruf wird mit diesem Feld die Nachrichten-ID der abzurufenden Nachricht angegeben. Normalerweise gibt der Warteschlangenmanager die erste Nachricht mit einer Nachrichten-ID und einer Korrelations-ID zurück, die mit den Feldern des Nachrichtendeskriptors übereinstimmen. Mit dem besonderen Wert MQC.MQMI_NONE können Sie zulassen, dass jede Nachrichten-ID übereinstimmt.

Bei einem MQQueue.Put-Aufruf wird mit diesem Feld die zu verwendende Nachrichten-ID angegeben. Wenn MQC.MQMI_NONE angegeben wurde, erstellt der Warteschlangenmanager eine eindeutige Nachrichten-ID, sobald die Nachricht eingereiht wird. Der Wert dieser Elementvariablen wird nach dem Einreihen aktualisiert, um anzugeben, welche Nachrichten-ID verwendet wurde. Der Standardwert ist MQC.MQMI_NONE.

public int MessageLength {get;}
Die Anzahl der Bytes der Nachrichtendaten im MQMessage-Objekt.

public int MessageSequenceNumber {get; set;}
Die Folgenummer einer logischen Nachricht in einer Gruppe.

public int MessageType {get; set;}
Gibt den Typ der Nachricht an. Folgende Werte sind derzeit vom System definiert:

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

Anwendungsdefinierte Werte können im Bereich zwischen MQC.MQMT_APPL_FIRST und MQC.MQMT_APPL_LAST ebenfalls verwendet werden. Der Standardwert für dieses Feld ist MQC.MQMT_DATAGRAM.

public int Offset {get; set;}
In einer segmentierten Nachricht ist dies die relative Position der Daten in einer physischen Nachricht ab Beginn einer logischen Nachricht.

public int OriginalLength {get; set;}
Die ursprüngliche Länge einer segmentierten Nachricht.

public int Persistence {get; set;}

Nachrichtenpersistenz. Die folgenden Werte sind definiert:

- MQC.MQPER_NOT_PERSISTENT

  Wenn Sie diese Option bei einem Client verwenden, für den die Verbindung wiederhergestellt werden kann, wird der Ursachencode MQRC_NONE an die Anwendung zurückgegeben, sobald die Verbindung erfolgreich hergestellt wurde.

- MQC.MQPER_PERSISTENT

  Wenn Sie diese Option bei einem Client verwenden, für den die Verbindung wiederhergestellt werden kann, wird der Ursachencode MQRC_CALL_INTERRUPTED an die Anwendung zurückgegeben, sobald die Verbindung erfolgreich hergestellt wurde.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

Der Standardwert ist MQC.MQPER_PERSISTENCE_AS_Q_DEF, d. h., die Persistenz für die Nachricht wird aus dem Standardpersistenzattribut der Zielwarteschlange übernommen.

public int Priority {get; set;}
Die Nachrichtenpriorität. Der besondere Wert MQC.MQPRI_PRIORITY_AS_Q_DEF kann auch bei abgehenden Nachrichten festgelegt werden. Die Priorität für die Nachricht wird dann aus dem Standardprioritätsattribut der Zielwarteschlange übernommen. Der Standardwert ist MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}
Gibt an, ob die Validierung von Eigenschaften stattfindet, wenn eine Eigenschaft der Nachricht festgelegt wird. Mögliche Werte:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

Der Standardwert ist MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}
Der Name der Anwendung, die die Nachricht eingereiht hat. Der Standardwert ist " ".

public int PutApplicationType {get; set;}
Die Art der Anwendung, von der die Nachricht eingereiht wurde. PutApplicationType kann ein systemdefinierter oder ein benutzerdefiniert Wert sein. Folgende Werte sind vom System definiert:

- 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

Der Standardwert ist MQC.MQAT_NO_CONTEXT, d. h. dass in der Nachricht keine Kontextinformationen vorhanden sind.

public DateTime PutDateTime {get; set;}
Das Datum mit Uhrzeit, an dem die Nachricht eingereiht wurde.

public string ReplyToQueueManagerName {get; set;}
Der Name des Warteschlangenmanagers, an den die Antwort- oder Berichtsnachrichten gesendet werden sollen. Der Standardwert ist "", und der Warteschlangenmanager stellt ReplyToQueueManagerName bereit.

public string ReplyToQueueName {get; set;}
Der Name der Nachrichtenwarteschlange, an die die Anwendung, von der die Abrufanforderung für die Nachricht ausgegeben wurde, die MQC.MQMT_REPLY- und MQC.MQMT_REPORT-Nachrichten senden soll. Der Standardwert für ReplyToQueueName ist "".

public int Report {get; set;}
Mit Report können Sie Optionen über Berichts- und Antwortnachrichten angeben:

- Sie können angeben, ob Berichte erforderlich sind.
- Sie können angeben, ob die Anwendungsnachrichtendaten in die Berichte eingeschlossen werden sollen.
- Sie können angeben, wie die Nachrichten- und Korrelations-IDs im Bericht oder in der Antwort festgelegt werden sollen.

Sie können eine beliebige Kombination aus den vier Berichtstypen anfordern:

- Geben Sie eine beliebige Kombination aus den vier Berichtstypen an. Wählen Sie dabei eine der drei Optionen für jeden Berichtstyp aus. Ihre Auswahl hängt davon ab, ob Sie die Anwendungsnachrichtendaten in die Berichtsnachricht einschließen möchten.

  1. Bestätigen bei Eingang
     - MQC.MQRO_COA
     - MQC.MQRO_COA_WITH_DATA
     - MQC.MQRO_COA_WITH_FULL_DATA**
  2. Empfangsbestätigung
     - MQC.MQRO_COD
     - MQC.MQRO_COD_WITH_DATA
     - MQC.MQRO_COD_WITH_FULL_DATA**
  3. Ausnahmebedingung
     - MQC.MQRO_EXCEPTION
     - MQC.MQRO_EXCEPTION_WITH_DATA
     - MQC.MQRO_EXCEPTION_WITH_FULL_DATA**
  4. Ablauf
     - MQC.MQRO_EXPIRATION
     - MQC.MQRO_EXPIRATION_WITH_DATA
     - MQC.MQRO_EXPIRATION_WITH_FULL_DATA**

  Anmerkung: Werte, die in der Liste mit ** markiert sind, werden von den z/OS-Warteschlangenmanagern nicht unterstützt. Verwenden Sie diese Werte nicht, wenn davon auszugehen ist, dass Ihre Anwendung auf einen z/OS-Warteschlangenmanager zugreift, und zwar unabhängig von der Plattform, auf der die Anwendung ausgeführt wird.

- Geben Sie eines der folgenden Elemente an, um zu steuern, wie die Nachrichten-ID für die Berichts- oder Antwortnachricht erstellt wird:
  - MQC.MQRO_NEW_MSG_ID
  - MQC.MQRO_PASS_MSG_ID
- Geben Sie eines der folgenden Elemente an, um zu steuern, wie die Korrelations-ID für die Berichts- oder Antwortnachricht festgelegt werden soll:
  - MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID
  - MQC.MQRO_PASS_CORREL_ID
- Geben Sie eines der folgenden Elemente an, um die Disposition der ursprünglichen Nachricht zu steuern, wenn diese nicht an die Zielwarteschlange übermittelt werden kann:
  - MQC.MQRO_DEAD_LETTER_Q
  - MQC.MQRO_DISCARD_MSG**
- Wenn keine Berichtsoptionen angegeben werden, gelten folgende Standardeinstellungen:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- Sie können eines der folgenden Elemente (oder beide) angeben, um anzufordern, dass die empfangende Anwendung eine Berichtsnachricht über eine positive oder negative Aktion sendet.
  - MQC.MQRO_PAN
  - MQC.MQRO_NAN

public int TotalMessageLength {get;}
Die Gesamtzahl der Bytes in der Nachricht, die in der Nachrichtenwarteschlange gespeichert ist, von der diese Nachricht empfangen wurde.

public string UserId {get; set;}
UserId ist Bestandteil des Identitätskontexts der Nachricht. Der Warteschlangenmanager stellt in der Regel den Wert bereit. Sie können den Wert überschreiben, wenn Sie die Berechtigung zum Festlegen des Identitätskontexts besitzen.

public int Version {get; set;}
Die Version der verwendeten MQMD-Struktur.

Die Nachrichtenmethoden Read und Write

Die Methoden Read und Write führen dieselben Funktionen wie die Elemente der Klassen BinaryReader und BinaryWriter im .NET-Namensbereich System.IO durch. Die vollständige Sprachsyntax und Nutzungsbeispiele finden Sie auf der MSDN-Website. Die Methoden lesen oder schreiben ab der aktuellen Position in den Nachrichtenpuffer. Sie verschieben die aktuelle Position um die Anzahl der gelesenen oder geschriebenen Bytes nach vorne.

Anmerkung: Wenn die Nachrichtendaten einen MQRFH- oder MQRFH2-Header enthalten, müssen Sie die Daten mithilfe der Methode ReadBytes lesen.

- Alle Methoden lösen eine E/A-Ausnahme (IOException) aus.
- Mit den ReadFully-Methoden wird automatisch die Größe der Zielfeldgruppe byte oder sbyte geändert, damit die Nachricht exakt passt. Auch die Größe eines Nullfelds wird geändert.
- Die Read-Methode löst EndOfStreamException aus.
- Die WriteDecimal-Methode löst MQException aus.
- Mit den Methoden ReadString, ReadLine und WriteString wird die Konvertierung zwischen Unicode und dem Zeichensatz der Nachricht vorgenommen; siehe unter CharacterSet.
- Mit den Decimal-Methoden werden gepackte Dezimalzahlen gelesen und geschrieben, die entweder im Big-Endian-Format, MQC.MQENC_DECIMAL_NORMAL, oder im Little-Endian-Format,

MQC.MQENC_DECIMAL_REVERSE, codiert werden. Das Format hängt vom Wert der Codierung (Encoding) ab. Es gibt folgende Dezimalstellenbereiche und die entsprechenden .NET-Typen:

Decimal2/short
-999 bis 999

Decimal4/int
-9999999 bis 9999999

Decimal8/long
-999999999999999 bis 999999999999999

- Mit den Methoden Double und Float werden Gleitkommazahlen gelesen und geschrieben, die im IEEE Big-Endian- oder Little-Endian-Format codiert sind (MQC.MQENC_FLOAT_IEEE_NORMAL bzw. MQC.MQENC_FLOAT_IEEE_REVERSED). Alternativ ist auch das S/390-Format, MQC.MQENC_FLOAT_S390, möglich. Das Format hängt vom Wert der Codierung (Encoding) ab.
- Mit den Int-Methoden werden Ganzzahlwerte gelesen und geschrieben, die entweder im Big-Endian-Format, MQC.MQENC_INTEGER_NORMAL, oder im Little-Endian-Format, MQC.MQENC_INTEGER_REVERSED, codiert werden. Das Format hängt vom Wert der Codierung (Encoding) ab. Die Ganzzahlen sind alle signiert, mit Ausnahme der Hinzufügung eines nicht signierten, aus 2 Bytes bestehenden ganzzahligen Typs. Die Ganzzahlgrößen sowie die .NET- und WebSphere MQ-Typen lauten wie folgt:

  2 Byte
  short, Int2, ushort, UInt2

  4 Byte
  int, Int4

  8 Byte
  long, Int8
- WriteObject überträgt die Klasse eines Objekts, die Werte der zugehörigen nicht transienten und nicht statischen Felder und die Felder der zugehörigen Supertypen in den Nachrichtenpuffer.
- ReadObject erstellt ein Objekt aus der Klasse des Objekts, der Signatur der Klasse und der Werte der zugehörigen nicht transienten und nicht statischen Felder sowie der Felder der zugehörigen Supertypen.

Tabelle 608. Nachrichtenmethoden für Lese- und Schreibvorgänge

Zieltyp	Methodensignaturen

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)`

Tabelle 608. Nachrichtenmethoden für Lese- und Schreibvorgänge (Forts.)

Zieltyp	Methodensignaturen
Decimal2	`public void WriteDecimal2(short value)`
Decimal4	`public void WriteDecimal4(short value)`
Decimal8	`public void WriteDecimal8(short value)`
Double	`public double ReadDouble()` `public void WriteDouble(double value)`
Float	`public float ReadFloat()` `public void WriteFloat(float value)`
Int2	`public void WriteInt2(int value)`

Int4	`public int readDecimal4()`
	`public int ReadInt()`
	`public int ReadInt4()`
	`public void WriteInt(int value)`
	`public void WriteInt4(int value)`

Int8	`public void WriteInt8(long value)`

Long	`public long ReadDecimal8()`
	`public long ReadLong()`
	`public long ReadInt8()`
	`public void WriteLong(long value)`

Object	`public Object ReadObject()` `public void WriteObject(Object object)`

Short	`public short ReadShort()`
	`public short ReadDecimal2()`
	`public short ReadInt2()`
	`public void WriteShort(int value)`

string	`public string ReadString(int length)` `public void WriteString(string string)`
Unsigned Short	`public ushort ReadUnsignedShort()` `public ushort ReadUInt2()`

Tabelle 608. Nachrichtenmethoden für Lese- und Schreibvorgänge (Forts.)	
Zieltyp	**Methodensignaturen**

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)`

Puffermethoden

`public void ClearMessage();`

Löst eine E/A-Ausnahme (`IOException`) aus.

Alle Daten im Nachrichtenpuffer werden verworfen, und die relative Datenadresse wird auf null zurückgesetzt.

`public void ResizeBuffer(int size)`

Löst eine E/A-Ausnahme (`IOException`) aus.

Hinweis auf das `MQMessage`-Objekt zur Puffergröße, die für nachfolgende Abrufvorgänge möglicherweise erforderlich ist. Wenn die Nachricht derzeit Nachrichtendaten enthält und die neue Größe kleiner ist als die aktuelle Größe, werden die Nachrichtendaten abgeschnitten.

`public void Seek(int pos)`

Löst die Ausnahmen `IOException`, `ArgumentOutOfRangeException`, `ArgumentException` aus.

Verschiebt den Cursor an die absolute Position im Nachrichtenpuffer, die mit *pos* angegeben wird. Nachfolgende Lese- und Schreibvorgänge agieren an dieser Position im Puffer.

`public int SkipBytes(int i)`

Löst die Ausnahmen `IOException`, `EndOfStreamException` aus.

Verschiebt n Bytes im Nachrichtenpuffer nach vorne und gibt n, die Anzahl übersprungener Bytes zurück.

Mit der Methode `SkipBytes` werden weitere Vorgänge blockiert, bis eines der folgenden Ereignisse eintritt:

- Alle Bytes wurden übersprungen.
- Das Ende des Nachrichtenpuffers wurde gefunden.
- Eine Ausnahme wird ausgelöst.

Eigenschaftenmethoden

`public void DeleteProperty(string name);`

Löst eine E/A-Ausnahme (`MQException`) aus.

Löscht eine Eigenschaft mit dem angegebenen Namen aus der Nachricht.

name
Der Name der zu löschenden Eigenschaft.

public System.Collections.IEnumerator GetPropertyNames(string *name*)

Löst eine E/A-Ausnahme (MQException) aus.

Gibt IEnumerator für alle Eigenschaftsnamen zurück, die mit dem angegebenen Namen übereinstimmen. Das Prozentzeichen '%' kann am Ende des Namens als Platzhalterzeichen verwendet werden, um die Eigenschaften der Nachricht bei Übereinstimmung mit null oder mehr Zeichen einschließlich des Punktes zu filtern.

name
Der Name der Eigenschaft für den Abgleich.

- Alle SetProperty- und GetProperty-Methoden lösen MQException aus.

Tabelle 609. SetProperty- und GetProperty-Methoden

Typ	Methodensignaturen

Boolean	`public boolean GetBooleanProperty(string name);`
	`public boolean GetBooleanProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBooleanProperty(string name, boolean value);`
	`public void SetBooleanProperty(string name, MQPropertyDescriptor pd, boolean value);`

Byte	`public sbyte GetByteProperty(string name);`
	`public sbyte GetByteProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetByteProperty(string name, sbyte value);`
	`public void SetByteProperty(string name, MQPropertyDescriptor pd, sbyte value);`

Bytes	`public sbyte[] GetBytesProperty(string name);`
	`public sbyte[] GetBytesProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBytesProperty(string name, sbyte[] value);`
	`public void SetBytesProperty(string name, MQPropertyDescriptor pd, sbyte[] value);`

Double	`public double GetDoubleProperty(string name);`
	`public double GetDoubleProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetDoubleProperty(string name, double value);`
	`public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);`

Float	`public float GetFloatProperty(string name);`
	`public float GetFloatProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetFloatProperty(string name, float value);`
	`public void SetFloatProperty(string name, MQPropertyDescriptor pd, float value);`

Int2	`public short GetInt2Property(string name);`
	`public short GetInt2Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt2Property(string name, short value);`
	`public void SetInt2Property(string name, MQPropertyDescriptor pd, short value);`

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);`

Tabelle 609. SetProperty- und GetProperty-Methoden (Forts.)

Typ	Methodensignaturen

Int8	`public long GetInt8Property(string name);`
	`public long GetInt8Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt8Property(string name, long value);`
	`public void SetInt8Property(string name, MQPropertyDescriptor pd, long value);`

Long	`public long GetLongProperty(string name);`
	`public long GetLongProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetLongProperty(string name, long value);`
	`public void SetLongProperty(string name, MQPropertyDescriptor pd, long value);`

Object	`public Object GetObjectProperty(string name);`
	`public Object GetObjectProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetObjectProperty(string name, Object value);`
	`public void SetObjectProperty(string name, MQPropertyDescriptor pd, Object value);`

Short	`public short GetShortProperty(string name);`
	`public short GetShortProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetShortProperty(string name, short value);`
	`public void SetShortProperty(string name, MQPropertyDescriptor pd, short value);`

string	`public string GetStringProperty(string name);`
	`public string GetStringProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetStringProperty(string name, string value);`
	`public void SetStringProperty(string name, MQPropertyDescriptor pd, string value);`

Konstruktoren

public MQMessage();
Erstellt ein MQMessage-Objekt mit den standardmäßigen Nachrichtendeskriptorinformationen und einem leeren Nachrichtenpuffer.

.NET-Klasse MQProcess

Mit MQProcess können Sie die Attribute eines WebSphere MQ-Prozesses abfragen. Erstellen Sie ein MQProcess-Objekt mithilfe eines Konstruktors oder einer MQQueueManager AccessProcess-Methode.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQProcess
```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- „Eigenschaften” auf Seite 1304
- „Konstruktoren” auf Seite 1304

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public string ApplicationId {get;}

Ruft die Zeichenfolge ab, mit der die zu startende Anwendung bestimmt wird. ApplicationId wird von einer Auslösemonitoranwendung verwendet. ApplicationId wird als Teil der Auslösenachricht an die Initialisierungswarteschlange gesendet.

Der Standardwert ist null.

public int ApplicationType {get;}

Gibt den Prozesstyp an, der von einer Auslösemonitoranwendung gestartet werden soll. Es sind bereits Standardtypen definiert, Sie können jedoch auch andere Typen verwenden:

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

Der Standardwert ist MQAT_NATIVE.

public string EnvironmentData {get;}

Ruft Informationen zur Umgebung der zu startenden Anwendung ab.

Der Standardwert ist null.

public string UserData {get;}

Ruft Informationen ab, die der Benutzer über die zu startende Anwendung bereitgestellt hat.

Der Standardwert ist null.

Konstruktoren

public MQProcess(MQQueueManager *queueManager*, string *processName*, int *openOptions*);
public MQProcess(MQQueueManager *qMgr*, string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Löst MQException aus.

Greifen Sie auf einen WebSphere MQ-Prozess im Warteschlangenmanager *qMgr* zu, um die Prozessattribute abzufragen.

qMgr
Warteschlangenmanager, auf den zugegriffen werden soll.

processName
Der Name des zu öffnenden Prozesses.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs des Prozesses. Folgende gültige Optionen können hinzugefügt oder mit einem bitweisen ODER kombiniert werden:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
Der Name des Warteschlangenmanagers, auf dem der Prozess definiert ist. Sie können einen leeren Warteschlangenmanagernamen oder einen Nullwert für den Namen angeben, wenn der Warteschlangenmanager mit dem übereinstimmt, auf den der Prozess zugreift.

alternateUserId

Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY im Parameter *openOptions* angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für die Aktion überprüft wird. Wenn MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

Die Standardbenutzerberechtigung wird für die Verbindung zum Warteschlangenmanager verwendet, wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist.

public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*);
public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Löst MQException aus.

Greifen Sie auf einen WebSphere MQ-Prozess in diesem Warteschlangenmanager zu, um die Prozessattribute abzufragen.

processName
Der Name des zu öffnenden Prozesses.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs des Prozesses. Folgende gültige Optionen können hinzugefügt oder mit einem bitweisen ODER kombiniert werden:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
Der Name des Warteschlangenmanagers, auf dem der Prozess definiert ist. Sie können einen leeren Warteschlangenmanagernamen oder einen Nullwert für den Namen angeben, wenn der Warteschlangenmanager mit dem übereinstimmt, auf den der Prozess zugreift.

alternateUserId

Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY im Parameter *openOptions* angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für die Aktion überprüft wird. Wenn MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

Die Standardbenutzerberechtigung wird für die Verbindung zum Warteschlangenmanager verwendet, wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist.

.NET-Klasse MQPropertyDescriptor

Verwenden Sie MQPropertyDescriptor als Parameter für MQMessage GetProperty- und SetProperty-Methoden. MQPropertyDescriptor beschreibt eine MQMessage-Eigenschaft.

Klasse

```
System.Object
        └ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```


- „Eigenschaften” auf Seite 1306
- „Konstruktoren” auf Seite 1307

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public int Context {get; set;}

Der Nachrichtenkontext, zu dem die Eigenschaft gehört. Mögliche Werte:

MQC.MQPD_NO_CONTEXT
Die Eigenschaft ist keinem Nachrichtenkontext zugeordnet.

MQC.MQPD_USER_CONTEXT
Die Eigenschaft wird dem Benutzerkontext zugeordnet.

Wenn der Benutzer über die entsprechenden Berechtigungen verfügt, wird eine dem Benutzerkontext zugeordnete Eigenschaft gespeichert, sobald eine Nachricht abgerufen wird. Eine nachfolgende Put -Methode, die auf den gespeicherten Kontext verweist, kann die Eigenschaft an die neue Nachricht übergeben.

public int CopyOptions {get; set;}

CopyOptions beschreibt, in welchen Nachrichtentyp die Eigenschaft kopiert werden kann.

Wenn ein Warteschlangenmanager eine Nachricht erhält, die eine mit WebSphere MQ definierte Eigenschaft enthält, die der Warteschlangenmanager als falsch erkennt, korrigiert der Warteschlangenmanager den Wert des Feldes CopyOptions.

Sie können eine beliebige Kombination der folgenden Optionen angeben. Kombinieren Sie die Optionen, indem Sie die Werte hinzufügen oder die bitweise ORverwenden.

MQC.MQCOPY_ALL
Die Eigenschaft wird in alle nachfolgenden Nachrichten kopiert.

MQC.MQCOPY_FORWARD
Die Eigenschaft wird in eine Nachricht kopiert, die weitergeleitet wird.

MQC.MQCOPY_PUBLISH
Die Eigenschaft wird in die Nachricht kopiert, die beim Veröffentlichen einer Nachricht von einem Subskribenten empfangen wird.

MQC.MQCOPY_REPLY
Die Eigenschaft wird in eine Antwortnachricht kopiert.

MQC.MQCOPY_REPORT
Die Eigenschaft wird in eine Berichtsnachricht kopiert.

MQC.MQCOPY_DEFAULT
Mit dem Wert wird angezeigt, dass keine anderen Kopieroptionen angegeben wurden. Zwischen der Eigenschaft und den nachfolgenden Nachrichten besteht keine Beziehung. MQC.MQCOPY_DEFAULT wird für Nachrichtendeskriptoreigenschaften immer zurückgegeben.

MQC.MQCOPY_NONE
Identisch mit MQC.MQCOPY_DEFAULT

public int Options { set; }
Options nimmt standardmäßig den Wert CMQC.MQPD_NONE an. Sie können keinen anderen Wert festlegen.

public int Support { get; set; }

Legen Sie Support fest, um die Unterstützungsstufe anzugeben, die für mit WebSphere MQ definierte Nachrichteneigenschaften erforderlich ist. Die Unterstützung für alle anderen Eigenschaften ist optional. Sie können einen beliebigen oder keinen der folgenden Werte angeben.

MQC.MQPD_SUPPORT_OPTIONAL
Die Eigenschaft wird von einem Warteschlangenmanager auch dann akzeptiert, wenn sie nicht unterstützt wird. Die Eigenschaft kann gelöscht werden, damit die Nachricht an einen Warteschlangenmanager weitergeleitet werden kann, der keine Nachrichteneigenschaften unterstützt. Dieser Wert wird auch Eigenschaften zugewiesen, die nicht in WebSphere MQ definiert sind.

MQC.MQPD_SUPPORT_REQUIRED
Die Unterstützung für die Eigenschaft ist erforderlich. Wenn Sie die Nachricht in einen Warteschlangenmanager einreihen, der keine Unterstützung für in WebSphere MQ definierte Eigenschaften bietet, schlägt die Methode fehl. Dabei werden der Beendigungscode MQC.MQCC_FAILED und der Ursachencode MQC.MQRC_UNSUPPORTED_PROPERTY zurückgegeben.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Die Unterstützung für die Eigenschaft ist erforderlich, wenn die Nachricht für eine lokale Warteschlange bestimmt ist. Wenn Sie die Nachricht in eine lokale Warteschlange auf einem Warteschlangenmanager einreihen, der keine Unterstützung für in WebSphere MQ definierte Eigenschaften bietet, schlägt die Methode fehl. Dabei werden der Beendigungscode MQC.MQCC_FAILED und der Ursachencode MQC.MQRC_UNSUPPORTED_PROPERTY zurückgegeben.

Beim Einreihen in einen fernen Warteschlangenmanager wird keine Überprüfung durchgeführt.

Konstruktoren

PropertyDescriptor();
Erstellt einen Eigenschaftsdeskriptor.

.NET-Klasse MQPutMessageOptions

Mit MQPutMessageOptions können Sie angeben, wie Nachrichten gesendet werden. Mit dieser Klasse wird das Verhalten von MQDestination.Put geändert.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- „Eigenschaften” auf Seite 1307„Konstruktoren” auf Seite 1310

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

Anmerkung: Das Verhalten einiger der in dieser Klasse verfügbaren Optionen hängt von der Umgebung ab, in der sie verwendet werden. Diese Elemente sind mit einem Stern (*) markiert.

public MQQueue ContextReference {get; set;}

Wenn das Feld `options` `MQC.MQPMO_PASS_IDENTITY_CONTEXT` oder `MQC.MQPMO_PASS_ALL_CONTEXT` enthält, legen Sie fest, dass dieses Feld auf die MQ-Warteschlange (`MQQueue`) verweist, aus der die Kontextinformationen übernommen werden sollen.

Der Anfangswert dieses Felds ist null.

public int InvalidDestCount {get;}*

Wird in der Regel für Verteilerlisten verwendet. `InvalidDestCount` gibt die Anzahl der Nachrichten an, die nicht an Warteschlangen in einer Verteilerliste gesendet werden konnten. Die Anzahl enthält Warteschlangen, die nicht geöffnet werden konnten, und Warteschlangen, die zwar erfolgreich geöffnet wurden, für die jedoch die PUT-Operation fehlgeschlagen ist.

.NET bietet keine Unterstützung für Verteilerlisten, `InvalidDestCount` wird jedoch beim Öffnen einer einzelnen Warteschlange festgelegt.

public int KnownDestCount {get;} *

Wird in der Regel für Verteilerlisten verwendet. `KnownDestCount` gibt die Anzahl der Nachrichten an, die der aktuelle Aufruf erfolgreich an Warteschlangen gesendet hat, die in lokale Warteschlangen aufgelöst werden.

.NET bietet keine Unterstützung für Verteilerlisten, `InvalidDestCount` wird jedoch beim Öffnen einer einzelnen Warteschlange festgelegt.

public int Options {get; set;}

Optionen, mit denen Sie die Aktion von `MQDestination.put` und `MQQueueManager.put` steuern können. Sie können einen beliebigen oder keinen der folgenden Werte angeben. Wenn mehrere Optionen erforderlich sind, können die Werte hinzugefügt oder mithilfe des bitweisen ODER-Operators kombiniert werden.

MQC.MQPMO_ASYNC_RESPONSE
Diese Option sorgt dafür, dass der `MQDestination.put`-Aufruf asynchron mit bestimmten Antwortdaten durchgeführt wird.

MQC.MQPMO_DEFAULT_CONTEXT
Ordnen Sie der Nachricht Standardkontext zu.

MQC.MQPMO_FAIL_IF_QUIESCING
Fehler, wenn der Warteschlangenmanager stillgelegt wird.

MQC.MQPMO_LOGICAL_ORDER*
Bringen Sie logische Nachrichten und Segmente in Nachrichtengruppen in ihre logische Reihenfolge.

Wenn Sie die Option `MQPMO_LOGICAL_ORDER` bei einem Client verwenden, für den die Verbindung wiederhergestellt werden kann, wird der Ursachencode `MQRC_RECONNECT_INCOMPATIBLE` an die Anwendung zurückgegeben.

MQC.MQPMO_NEW_CORREL_ID*
Generieren Sie eine neue Korrelations-ID für jede gesendete Nachricht.

MQC.MQPMO_NEW_MSG_ID*
Generieren Sie eine neue Nachrichten-ID für jede gesendete Nachricht.

MQC.MQPMO_NONE
Keine Optionen angegeben. Diese Option darf nicht mit anderen Optionen verwendet werden.

MQC.MQPMO_NO_CONTEXT
Kein der Nachricht zuzuordnender Kontext vorhanden.

MQC.MQPMO_NO_SYNCPOINT
Nachricht ohne Synchronisationspunktsteuerung verwenden. Wenn die Option für die Synchronisationspunktsteuerung nicht angegeben ist, wird als Standardeinstellung angenommen, dass kein Synchronisationspunkt vorhanden ist.

MQC.MQPMO_PASS_ALL_CONTEXT
Gesamten Kontext von einer Eingabe-WS-Kennung übergeben.

MQC.MQPMO_PASS_IDENTITY_CONTEXT
Identitätskontext von einer Eingabe-WS-Kennung übergeben.

MQC.MQPMO_RESPONSE_AS_Q_DEF

Bei einem `MQDestination.put`-Aufruf übernimmt diese Option den Put-Antworttyp aus dem Attribut DEFPRESP der Warteschlange.

Bei einem `MQQueueManager.put`-Aufruf wird bei Angabe dieser Option der Aufruf synchron durchgeführt.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF
`MQC.MQPMO_RESPONSE_AS_TOPIC_DEF` ist ein Synonym für `MQC.MQPMO_RESPONSE_AS_Q_DEF` zur Verwendung mit Themenobjekten.

MQC.MQPMO_RETAIN

Die gesendete Veröffentlichung muss vom Warteschlangenmanager als ständige Veröffentlichung bereitgestellt werden. Wenn diese Option angegeben, aber eine ständige Veröffentlichung nicht möglich ist, wird die betreffende Nachricht nicht veröffentlicht und der Aufruf schlägt mit `MQC.MQRC_PUT_NOT_RETAINED` fehl.

Fordern Sie eine Kopie dieser Veröffentlichung nach der Veröffentlichung an, indem Sie die Methode `MQSubscription.RequestPublicationUpdate` aufrufen. Die gespeicherte Veröffentlichung wird an Anwendungen gesendet, die dann eine Subskription erstellen, ohne die Option `MQC.MQSO_NEW_PUBLICATIONS_ONLY` festzulegen. Überprüfen Sie die `MQIsRetained`-Nachrichteneigenschaft einer Veröffentlichung, wenn diese eingeht, um festzustellen, ob es sich um eine ständige Veröffentlichung handelt.

Wenn ständige Veröffentlichungen von einem Subskribenten angefordert werden, kann die Subskription einen Platzhalter in der Themenzeichenfolge aufweisen. Wenn mehrere ständige Veröffentlichungen in der Themenstruktur vorliegen, die mit der Subskription übereinstimmen, werden alle gesendet.

MQC.MQPMO_SET_ALL_CONTEXT
Den gesamten Kontext über die Anwendung festlegen.

MQC.MQPMO_SET_IDENTITY_CONTEXT
Den Identitätskontext über die Anwendung festlegen.

MQC.MQPMO_SYNC_RESPONSE
Diese Option sorgt dafür, dass der `MQDestination.put`- oder `MQQueueManager.put`-Aufruf synchron mit den gesamten Antwortdaten durchgeführt wird.

MQC.MQPMO_SUPPRESS_REPLYTO
Alle in den Feldern `ReplyToQueueName` und `ReplyToQueueManagerName` der Veröffentlichung eingetragenen Informationen werden nicht an Subskribenten weitergegeben. Wenn diese Option zusammen mit einer Berichtsoption, die einen `ReplyToQueueName`-Wert erfordert, verwendet wird, schlägt der Aufruf mit `MQC.MQRC_MISSING_REPLY_TO_Q` fehl.

MQC.MQPMO_SYNCPOINT
Nachricht mit Synchronisationspunktsteuerung verwenden. Die Nachricht wird erst außerhalb der Arbeitseinheit sichtbar, wenn die Arbeitseinheit festgeschrieben wird. Wird die Arbeitseinheit zurückgesetzt, wird die Nachricht gelöscht.

`public int RecordFields {get; set;}*`

Informationen über Verteilerlisten. Verteilerlisten werden in .NET nicht unterstützt.

public string ResolvedQueueManagerName {get;}

Ein Ausgabefeld, das vom Warteschlangenmanager auf den Namen des Warteschlangenmanagers gesetzt wurde, zu dem die Warteschlange gehört, die mit dem fernen Warteschlangennamen angegeben wurde. `ResolvedQueueManagerName` kann sich vom Namen des Warteschlangenmanagers unterscheiden, über den auf die Warteschlange zugegriffen wird, wenn es sich um eine ferne Warteschlange handelt.

Ein belegter Wert wird nur dann zurückgegeben, wenn es sich bei dem Objekt um eine einzelne Warteschlange handelt. Wenn sich das Objekt in einer Verteilerliste oder in einem Thema befindet, ist der zurückgegebene Wert nicht definiert.

public string ResolvedQueueName {get;}

Ein Ausgabefeld, das vom Warteschlangenmanager auf den Namen der Warteschlange gesetzt wurde, in dem die Nachricht abgelegt wurde. `ResolvedQueueName` unterscheidet sich möglicherweise von dem Namen, der zum Öffnen der Warteschlange verwendet wurde, wenn es sich bei der geöffneten Warteschlange um eine Aliaswarteschlange oder Modellwarteschlange handelt.

Ein belegter Wert wird nur dann zurückgegeben, wenn es sich bei dem Objekt um eine einzelne Warteschlange handelt. Wenn sich das Objekt in einer Verteilerliste oder in einem Thema befindet, ist der zurückgegebene Wert nicht definiert.

public int UnknownDestCount {get;} *

Das Feld `UnknownDestCount` wird in der Regel für Verteilerlisten verwendet und stellt ein Ausgabefeld dar, das vom Warteschlangenmanager festgelegt wird. Es gibt die Anzahl der Nachrichten an, die der aktuelle Aufruf erfolgreich an Warteschlangen gesendet hat, die in ferne Warteschlangen aufgelöst werden.

.NET bietet keine Unterstützung für Verteilerlisten, `InvalidDestCount` wird jedoch beim Öffnen einer einzelnen Warteschlange festgelegt.

Konstruktoren

public MQPutMessageOptions();

Erstellen Sie ein neues `MQPutMessageOptions`-Objekt ohne festgelegte Optionen und mit leeren Optionen `ResolvedQueueName` und `ResolvedQueueManagerName`.

.NET-Klasse MQQueue

Mit `MQQueue` können Sie Nachrichten senden und empfangen sowie Attribute einer WebSphere MQ-Warteschlange abfragen. Erstellen Sie ein `MQQueue`-Objekt mithilfe eines Konstruktors oder einer `MQQueueManager.AccessProcess`-Methode.

Klasse

```
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;`

- „Eigenschaften” auf Seite 1311
- „Methoden” auf Seite 1313

- „Konstruktoren” auf Seite 1316

Eigenschaften

Test für den Fall, dass `MQException` beim Abrufen von Eigenschaften ausgelöst wird.

public int ClusterWorkLoadPriority {get;}
Gibt die Priorität der Warteschlange an. Dieser Parameter ist nur für lokale und ferne Warteschlangen sowie für Aliaswarteschlangen gültig.

public int ClusterWorkLoadRank {get;}
Gibt die Rangfolge der Warteschlange an. Dieser Parameter ist nur für lokale und ferne Warteschlangen sowie für Aliaswarteschlangen gültig.

public int ClusterWorkLoadUseQ {get;}
Gibt das Verhalten einer MQPUT-Operation an, wenn die Zielwarteschlange eine lokale Instanz und mindestens eine ferne Clusterinstanz besitzt. Dieser Parameter gilt nicht, wenn der MQPUT-Aufruf von einem Clusterkanal ausgeht. Dieser Parameter ist nur für lokale Warteschlangen gültig.

public DateTime CreationDateTime {get;}
Datum und Uhrzeit der Erstellung dieser Warteschlange.

public int CurrentDepth {get;}
Ruft die Anzahl der Nachrichten ab, die sich derzeit in der Warteschlange befinden. Dieser Wert erhöht sich während eines PUT-Aufrufs und beim Zurücksetzen eines GET-Aufrufs. Der Wert wird niedriger, wenn Sie einen Abrufvorgang (GET, außer Anzeige) oder eine Rücksetzung eines PUT-Aufrufs durchführen.

public int DefinitionType {get;}
Angabe darüber, wie die Warteschlange definiert wurde. Folgende Werte sind möglich:

- `MQC.MQQDT_PREDEFINED`
- `MQC.MQQDT_PERMANENT_DYNAMIC`
- `MQC.MQQDT_TEMPORARY_DYNAMIC`

public int InhibitGet {get; set;}
Steuert, ob Sie Nachrichten aus dieser Warteschlange oder für dieses Thema abrufen können. Folgende Werte sind möglich:

- `MQC.MQQA_GET_INHIBITED`
- `MQC.MQQA_GET_ALLOWED`

public int InhibitPut {get; set;}
Steuert, ob Sie Nachrichten in diese Warteschlange oder für dieses Thema einreihen können. Folgende Werte sind möglich:

- `MQQA_PUT_INHIBITED`
- `MQQA_PUT_ALLOWED`

public int MaximumDepth {get;}
Die maximale Anzahl an Nachrichten, die gleichzeitig in der Warteschlange vorhanden sein können. Ein Versuch, eine Nachricht in eine Warteschlange einzureihen, die bereits diese Anzahl an Nachrichten enthält, schlägt mit dem Ursachencode `MQC.MQRC_Q_FULL` fehl.

public int MaximumMessageLength {get;}
Die maximale Länge der Anwendungsdaten, die in jeder Nachricht in dieser Warteschlange vorhanden sein kann. Ein Versuch, eine Nachricht einzureihen, die größer ist als dieser Wert, schlägt mit dem Ursachencode `MQC.MQRC_MSG_TOO_BIG_FOR_Q` fehl.

public int NonPersistentMessageClass {get;}
Die Zuverlässigkeitsebene für nicht persistente Nachrichten, die in diese Warteschlange eingereiht wurden.

public int OpenInputCount {get;}
Die Anzahl der Kennungen, die derzeit für das Entfernen von Nachrichten aus der Warteschlange gültig sind. `OpenInputCount` gibt die Gesamtzahl gültiger Eingabekennungen an, die dem lokalen Warteschlangenmanager bekannt sind. Es werden nicht nur die von der Anwendung erstellten Kennungen angegeben.

public int OpenOutputCount {get;}
Die Anzahl der Kennungen, die derzeit für das Hinzufügen von Nachrichten zur Warteschlange gültig sind. `OpenOutputCount` gibt die Gesamtzahl gültiger Ausgabekennungen an, die dem lokalen Warteschlangenmanager bekannt sind. Es werden nicht nur die von der Anwendung erstellten Kennungen angegeben.

public int QueueAccounting {get;}
Gibt an, ob Sie die Erfassung von Abrechnungsdaten für die Warteschlange aktivieren können.

public int QueueMonitoring {get;}
Gibt an, ob Sie die Überwachung für die Warteschlange aktivieren können.

public int QueueStatistics {get;}
Gibt an, ob Sie die Erfassung von Statistikdaten für die Warteschlange aktivieren können.

public int QueueType {get;}
Der Typ dieser Warteschlange mit einem der folgenden Werte:

- `MQC.MQQT_ALIAS`
- `MQC.MQQT_LOCAL`
- `MQC.MQQT_REMOTE`
- `MQC.MQQT_CLUSTER`

public int Shareability {get;}
Gibt an, ob die Warteschlange mehrmals für die Eingabe geöffnet werden kann. Folgende Werte sind möglich:

- `MQC.MQQA_SHAREABLE`
- `MQC.MQQA_NOT_SHAREABLE`

public string TPIPE {get;}
Der TPIPE-Name, der für die Kommunikation mit OTMA über WebSphere MQ IMS Bridge verwendet wird.

public int TriggerControl {get; set;}
Gibt an, ob Auslösenachrichten in eine Initialisierungswarteschlange geschrieben werden, um eine Anwendung zu starten, die einen Service für die Warteschlange bereitstellt. Folgende Werte sind möglich:

- `MQC.MQTC_OFF`
- `MQC.MQTC_ON`

public string TriggerData {get; set;}
Die Daten im freien Format, die der Warteschlangenmanager in die Auslösenachricht einfügt. Auslöserdaten (`TriggerData`) werden eingefügt, wenn eine in dieser Warteschlange eingehende Nachricht verursacht, dass eine Auslösenachricht in die Initialisierungswarteschlange geschrieben wird. Die maximal zulässige Länge der Zeichenfolge wird durch `MQC.MQ_TRIGGER_DATA_LENGTH` festgelegt.

public int TriggerDepth {get; set;}
Die Anzahl der Nachrichten, die sich in der Warteschlange befinden müssen, damit eine Auslösenachricht geschrieben wird (wenn der Auslösertyp auf `MQC.MQTT_DEPTH` gesetzt ist).

public int TriggerMessagePriority {get; set;}
Die Nachrichtenpriorität, unterhalb der Nachrichten nicht zur Erzeugung von Auslösenachrichten beitragen. Das bedeutet, der Warteschlangenmanager ignoriert diese Nachrichten bei der Entscheidung, ob ein Auslöser generiert werden soll. Mit einem Nullwert wird festgelegt, dass alle Nachrichten zur Erzeugung von Auslösenachrichten beitragen.

public int TriggerType {get; set;}

Die Bedingungen, unter denen Auslösenachrichten als Ergebnis von Nachrichten geschrieben werden, die in dieser Warteschlange eingehen. Folgende Werte sind möglich:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT_EVERY
- MQC.MQTT_DEPTH

Methoden

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Löst MQException aus.

Ruft eine Nachricht aus einer Warteschlange ab.

Wenn der Abrufvorgang fehlschlägt, bleibt das MQMessage-Objekt unverändert. Wenn der Vorgang erfolgreich ist, werden der Nachrichtendeskriptor und Teile der Nachrichtendaten von MQMessage durch den Nachrichtendeskriptor und die Nachrichtendaten aus der eingehenden Nachricht ersetzt.

Alle Aufrufe für WebSphere MQ von einem bestimmten MQQueueManager sind synchron. Wenn Sie daher einen Abruf mit Wartestatus durchführen, werden alle anderen Threads, die denselben MQQueueManager verwenden, blockiert und können keine weiteren WebSphere MQ-Aufrufe durchführen, bis der GET-Aufruf abgeschlossen ist. Wenn Sie mehrere Threads benötigen, um gleichzeitig auf WebSphere MQ zuzugreifen, muss jeder Thread sein eigenes MQQueueManager-Objekt erstellen.

message
Enthält der Nachrichtendeskriptor und die zurückgegebenen Nachrichtendaten. Einige Felder im Nachrichtendeskriptor sind Eingabeparameter. Es ist wichtig, sicherzustellen, dass die MessageId- und CorrelationId-Eingabeparameter als erforderlich festgelegt werden.

Ein Client, für den die Verbindung wiederhergestellt werden kann, gibt nach der erfolgreichen Verbindungswiederholung den Ursachencode MQRC_BACKED_OUT für Nachrichten zurück, die unter MQGM_SYNCPOINT empfangen wurden.

getMessageOptions
Optionen zum Steuern der Aktionen des Abrufs.

Die Verwendung von MQC.MQGMO_CONVERT führt möglicherweise zu einer Ausnahme mit dem Ursachencode MQC.MQRC_CONVERTED_STRING_TOO_BIG, wenn Sie eine Konvertierung von Codes mit Einzelbytezeichen in Codes mit Doppelbytezeichen durchführen. In diesem Fall wird die Nachricht ohne Konvertierung in den Zwischenspeicher kopiert.

Wenn *getMessageOptions* nicht angegeben wird, lautet die verwendete Nachrichtenoption MQGMO_NOWAIT.

Wenn Sie die Option MQGMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

MaxMsgSize
Die größte Nachricht, die dieses Nachrichtenobjekt erhalten darf. Wenn die Nachricht in der Warteschlange größer ist als dieser Wert, tritt eine der beiden folgenden Situationen ein:

- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag im MQGetMessageOptions-Objekt gesetzt ist, werden möglichst viele Nachrichtendaten in die Nachricht übernommen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_ACCEPTED ausgelöst.

- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag nicht gesetzt wird, wird die Nachricht in der Warteschlange belassen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_FAILED ausgelöst.

Wenn *MaxMsgSize* nicht angegeben ist, wird die gesamte Nachricht abgerufen.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Löst MQException aus.

Reiht eine Nachricht in eine Warteschlange ein.

Änderungen am MQMessage-Objekt, die nach Abschluss des PUT-Aufrufs vorgenommen werden, wirken sich nicht auf die tatsächliche Nachricht in der WebSphere MQ-Warteschlange oder im Veröffentlichungsthema aus.

Mit Put werden die MessageId- und CorrelationId-Eigenschaften des MQMessage-Objekts aktualisiert. Dabei werden keine Nachrichtendaten gelöscht. Weitere Put- oder Get-Aufrufe beziehen sich auf die aktualisierten Informationen im MQMessage-Objekt. Beispiel: Im folgenden Codeausschnitt enthält die erste Nachricht a und die zweite ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Ein MQMessage-Objekt, das die Nachrichtendeskriptordaten und die zu sendende Nachricht enthält. Der Nachrichtendeskriptor kann sich aufgrund dieser Methode ändern. Die Werte im Nachrichtendeskriptor sofort nach Abschluss dieser Methode sind die Werte, die in die Warteschlange eingereiht oder für das Thema veröffentlicht wurden.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- MQRC_CALL_INTERRUPTED, wenn die Verbindung unterbrochen wird, während ein PUT-Aufruf für eine persistente Nachricht durchgeführt wird und die Verbindungswiederholung erfolgreich ist.
- MQRC_NONE, wenn die Verbindung erfolgreich hergestellt wurde, während ein PUT-Aufruf für eine nicht persistente Nachricht durchgeführt wurde (siehe Anwendungswiederherstellung).

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessageOptons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

Anmerkung: Wenn Sie aus Gründen der Einfachheit und Leistungsoptimierung eine einzelne Nachricht in eine Warteschlange einreihen möchten, verwenden Sie das MQQueueManager.Put-Objekt. Hierfür sollten Sie über ein MQQueue-Objekt verfügen.

public void PutForwardMessage(MQMessage *message*);
public void PutForwardMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Löst MQException aus.

Reiht eine Nachricht ein, die gerade an die Warteschlange weitergeleitet wird. Dabei ist *message* die ursprüngliche Nachricht.

message
Ein MQMessage-Objekt, das die Nachrichtendeskriptordaten und die zu sendende Nachricht enthält. Der Nachrichtendeskriptor kann sich aufgrund dieser Methode ändern. Die Werte im Nach-

richtendeskriptor sofort nach Abschluss dieser Methode sind die Werte, die in die Warteschlange eingereiht oder für das Thema veröffentlicht wurden.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- MQRC_CALL_INTERRUPTED, wenn die Verbindung unterbrochen wird, während ein PUT-Aufruf für eine persistente Nachricht durchgeführt wird und die Verbindungswiederholung erfolgreich ist.
- MQRC_NONE, wenn die Verbindung erfolgreich hergestellt wurde, während ein PUT-Aufruf für eine nicht persistente Nachricht durchgeführt wurde (siehe Anwendungswiederherstellung).

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessageOptons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

public void PutReplyMessage(MQMessage *message*)
public void PutReplyMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Löst MQException aus.

Reiht eine Antwortnachricht in die Warteschlange ein. Dabei ist *message* die ursprüngliche Nachricht.

message
Enthält der Nachrichtendeskriptor und die zurückgegebenen Nachrichtendaten. Einige Felder im Nachrichtendeskriptor sind Eingabeparameter. Es ist wichtig, sicherzustellen, dass die MessageId- und CorrelationId-Eingabeparameter als erforderlich festgelegt werden.

Ein Client, für den die Verbindung wiederhergestellt werden kann, gibt nach der erfolgreichen Verbindungswiederholung den Ursachencode MQRC_BACKED_OUT für Nachrichten zurück, die unter MQGM_SYNCPOINT empfangen wurden.

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessageOptons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

public void PutReportMessage(MQMessage *message*)
public void PutReportMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Löst MQException aus.

Reiht eine Berichtsnachricht in die Warteschlange ein. Dabei ist *message* die ursprüngliche Nachricht.

message
Enthält der Nachrichtendeskriptor und die zurückgegebenen Nachrichtendaten. Einige Felder im Nachrichtendeskriptor sind Eingabeparameter. Es ist wichtig, sicherzustellen, dass die MessageId- und CorrelationId-Eingabeparameter als erforderlich festgelegt werden.

Ein Client, für den die Verbindung wiederhergestellt werden kann, gibt nach der erfolgreichen Verbindungswiederholung den Ursachencode MQRC_BACKED_OUT für Nachrichten zurück, die unter MQGM_SYNCPOINT empfangen wurden.

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessageOptons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

Konstruktoren

public MQQueue MQQueueManager.AccessQueue(string *queueName*, int *openOptions*);
public MQQueue MQQueueManager.AccessQueue(string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Löst MQException aus.

Greift auf eine Warteschlange in diesem Warteschlangenmanager zu.

Sie können Nachrichten abrufen, durchsuchen oder einreihen und die Attribute der Warteschlange abfragen oder festlegen. Wenn die benannte Warteschlange eine Modellwarteschlange ist, wird eine dynamische lokale Warteschlange erstellt. Fragen Sie das name-Attribut des resultierenden MQQueue-Objekts ab, um den Namen der dynamischen Warteschlange zu ermitteln.

queueName
Name der zu öffnenden Warteschlange.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs der Warteschlange.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Die Prüfung wird mit der angegebenen Benutzer-ID durchgeführt.

MQC.MQOO_BIND_AS_QDEF
Standardmäßige Bindung für Warteschlange verwenden.

MQC.MQOO_BIND_NOT_FIXED
Keine Bindung an ein bestimmtes Ziel.

MQC.MQOO_BIND_ON_OPEN
Bindung der Kennung an das Ziel, wenn die Warteschlange geöffnet wird.

MQC.MQOO_BROWSE
Zum Durchsuchen der Nachricht öffnen.

MQC.MQOO_FAIL_IF_QUIESCING
Fehler, wenn der Warteschlangenmanager stillgelegt wird.

MQC.MQOO_INPUT_AS_Q_DEF
Öffnen zum Empfangen von Nachrichten unter Verwendung der von der Warteschlange definierten Standardwerte.

MQC.MQOO_INPUT_SHARED
Öffnen zum Empfangen von Nachrichten mit gemeinsamem Zugriff.

MQC.MQOO_INPUT_EXCLUSIVE
Öffnen zum Empfangen von Nachrichten mit exklusivem Zugriff.

MQC.MQOO_INQUIRE
Öffnen für die Abfrage (erforderlich, wenn Sie Eigenschaften abfragen möchten).

MQC.MQOO_OUTPUT
Öffnen zum Einreihen von Nachrichten.

MQC.MQOO_PASS_ALL_CONTEXT
Weitergabe des gesamten Kontexts erlaubt.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Weitergabe des Identitätskontexts erlaubt.

MQC.MQOO_SAVE_ALL_CONTEXT
Speichern des Kontexts beim Abrufen einer Nachricht.

MQC.MQOO_SET
Öffnen zum Festlegen von Attributen (erforderlich, wenn Sie Eigenschaften festlegen möchten).

MQC.MQOO_SET_ALL_CONTEXT
Zulassen, dass der gesamte Kontext festgelegt wird.

MQC.MQOO_SET_IDENTITY_CONTEXT
Zulassen, dass der Identitätskontext festgelegt wird.

queueManagerName
Name des Warteschlangenmanagers, für den die Warteschlange definiert ist. Mit einem Namen, der völlig leer ist oder ausschließlich Nullwerte enthält, wird der Warteschlangenmanager angegeben, mit dem das MQQueueManager-Objekt verbunden ist.

dynamicQueueName
dynamicQueueName wird ignoriert, falls nicht mit queueName der Name einer Modellwarteschlange angegeben ist. In diesem Fall gibt *dynamicQueueName* den Namen der zu erstellenden dynamischen Warteschlange an. Ein leerer Name oder Nullname ist nicht gültig, wenn queueName den Namen einer Modellwarteschlange angibt. Wenn das letzte belegte Zeichen im Namen ein Stern (*) ist, ersetzt der Warteschlangenmanager den Stern durch eine Zeichenfolge. Mit den Zeichen wird sichergestellt, dass der für die Warteschlange generierte Name in diesem Warteschlangenmanager eindeutig ist.

alternateUserId
Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY im Parameter openOptions angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für den Öffnungsvorgang überprüft wird. Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

public MQQueue(MQQueueManager *queueManager*, string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Löst MQException aus.

Greift auf eine Warteschlange im Warteschlangenmanager (queueManager) zu.

Sie können Nachrichten abrufen, durchsuchen oder einreihen und die Attribute der Warteschlange abfragen oder festlegen. Wenn die benannte Warteschlange eine Modellwarteschlange ist, wird eine dynamische lokale Warteschlange erstellt. Fragen Sie das name-Attribut des resultierenden MQQueue-Objekts ab, um den Namen der dynamischen Warteschlange zu ermitteln.

queueManager
Warteschlangenmanager für den Zugriff auf die Warteschlange.

queueName
Name der zu öffnenden Warteschlange.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs der Warteschlange.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Die Prüfung wird mit der angegebenen Benutzer-ID durchgeführt.

MQC.MQOO_BIND_AS_QDEF
Standardmäßige Bindung für Warteschlange verwenden.

MQC.MQOO_BIND_NOT_FIXED
Keine Bindung an ein bestimmtes Ziel.

MQC.MQOO_BIND_ON_OPEN
Bindung der Kennung an das Ziel, wenn die Warteschlange geöffnet wird.

MQC.MQOO_BROWSE
Zum Durchsuchen der Nachricht öffnen.

MQC.MQOO_FAIL_IF_QUIESCING
Fehler, wenn der Warteschlangenmanager stillgelegt wird.

MQC.MQOO_INPUT_AS_Q_DEF
Öffnen zum Empfangen von Nachrichten unter Verwendung der von der Warteschlange definierten Standardwerte.

MQC.MQOO_INPUT_SHARED
Öffnen zum Empfangen von Nachrichten mit gemeinsamem Zugriff.

MQC.MQOO_INPUT_EXCLUSIVE
Öffnen zum Empfangen von Nachrichten mit exklusivem Zugriff.

MQC.MQOO_INQUIRE
Öffnen für die Abfrage (erforderlich, wenn Sie Eigenschaften abfragen möchten).

MQC.MQOO_OUTPUT
Öffnen zum Einreihen von Nachrichten.

MQC.MQOO_PASS_ALL_CONTEXT
Weitergabe des gesamten Kontexts erlaubt.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Weitergabe des Identitätskontexts erlaubt.

MQC.MQOO_SAVE_ALL_CONTEXT
Speichern des Kontexts beim Abrufen einer Nachricht.

MQC.MQOO_SET
Öffnen zum Festlegen von Attributen (erforderlich, wenn Sie Eigenschaften festlegen möchten).

MQC.MQOO_SET_ALL_CONTEXT
Zulassen, dass der gesamte Kontext festgelegt wird.

MQC.MQOO_SET_IDENTITY_CONTEXT
Zulassen, dass der Identitätskontext festgelegt wird.

queueManagerName
Name des Warteschlangenmanagers, für den die Warteschlange definiert ist. Mit einem Namen, der völlig leer ist oder ausschließlich Nullwerte enthält, wird der Warteschlangenmanager angegeben, mit dem das `MQQueueManager`-Objekt verbunden ist.

dynamicQueueName
dynamicQueueName wird ignoriert, falls nicht mit `queueName` der Name einer Modellwarteschlange angegeben ist. In diesem Fall gibt *dynamicQueueName* den Namen der zu erstellenden dynamischen Warteschlange an. Ein leerer Name oder Nullname ist nicht gültig, wenn `queueName` den Namen einer Modellwarteschlange angibt. Wenn das letzte belegte Zeichen im Namen ein Stern (`*`) ist, ersetzt der Warteschlangenmanager den Stern durch eine Zeichenfolge. Mit den Zeichen wird sichergestellt, dass der für die Warteschlange generierte Name in diesem Warteschlangenmanager eindeutig ist.

alternateUserId
Wenn `MQC.MQOO_ALTERNATE_USER_AUTHORITY` im Parameter `openOptions` angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für den Öffnungsvorgang überprüft wird. Wenn `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

.NET-Klasse `MQQueueManager`

Mit `MQQueueManager` können Sie eine Verbindung zu einem Warteschlangenmanager herstellen und auf die Objekte des Warteschlangenmanagers zugreifen. Mit der Klasse werden zudem Transaktionen gesteuert. Der `MQQueueManager`-Konstruktor erstellt entweder eine Client- oder eine Serververbindung.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.ManagedObject
                └ IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```


- „Eigenschaften” auf Seite 1319
- „Methoden” auf Seite 1322
- „Konstruktoren” auf Seite 1329

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public int AccountingConnOverride {get;}
Gibt an, ob Anwendungen die Einstellung für die MQI-Abrechnung und die Abrechnungswerte der Warteschlange überschreiben können.

public int AccountingInterval {get;}
Das Intervall, in dem temporäre Abrechnungssätze geschrieben werden (in Sekunden).

public int ActivityRecording {get;}
Steuert die Erzeugung von Aktivitätenberichten.

public int AdoptNewMCACheck {get;}
Gibt an, welche Elemente überprüft werden, um festzustellen, ob der Nachrichtenkanalagent angenommen wird, falls ein neuer eingehender Kanal erkannt wird. Damit der Agent angenommen wird, muss der Name des Nachrichtenkanalagenten mit dem Namen eines aktiven Nachrichtenkanalagenten übereinstimmen.

public int AdoptNewMCAInterval {get;}
Zeit in Sekunden, die der neue Kanal wartet, bis der verwaiste Kanal geschlossen wird.

public int AdoptNewMCAType {get;}
Gibt an, ob die verwaiste Instanz eines Nachrichtenkanalagenten angenommen (erneut gestartet) werden soll, wenn eine neue eingehende Kanalanforderung festgestellt wird, die mit dem Wert AdoptNewMCACheck übereinstimmt.

public int BridgeEvent {get;}
Gibt an, ob IMS Bridge-Ereignisse generiert werden sollen.

public int ChannelEvent {get;}
Gibt an, ob Kanalereignisse generiert werden sollen.

public int ChannelInitiatorControl {get;}
Gibt an, ob der Kanalinitiator automatisch startet, wenn der Warteschlangenmanager startet.

public int ChannelInitiatorAdapters {get;}
Die Anzahl der Adaptersubtasks für die Verarbeitung von WebSphere MQ-Aufrufen.

public int ChannelInitiatorDispatchers {get;}
Die Anzahl an Dispatchern, die für den Kanalinitiator verwendet werden sollen.

public int ChannelInitiatorTraceAutoStart {get;}
Gibt an, ob der Kanalinitiatortrace automatisch startet.

public int ChannelInitiatorTraceTableSize {get;}
Die Größe des Tracedatenbereichs eines -Kanalinitiators in Megabyte.

public int ChannelMonitoring {get;}
Gibt an, ob die Kanalüberwachung verwendet wird.

public int ChannelStatistics {get;}
Steuert die Erfassung von statistischen Daten für Kanäle.

public int CharacterSet {get;}
Gibt die ID des codierten Zeichensatzes (CCSID) für den Warteschlangenmanager zurück. `CharacterSet` wird vom Warteschlangenmanager für alle Zeichenfolgenfelder in der Anwendungsprogrammierschnittstelle (API) verwendet.

public int ClusterSenderMonitoring {get;}
Steuert die Erfassung von Onlineüberwachungsdaten für automatisch definierte Clustersenderkanäle.

public int ClusterSenderStatistics {get;}
Steuert die Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle.

public int ClusterWorkLoadMRU {get;}
Die maximale Anzahl der abgehenden Clusterkanäle.

public int ClusterWorkLoadUseQ {get;}
Der Standardwert der `MQQueue`-Eigenschaft, `ClusterWorkLoadUseQ`, wenn damit ein Wert von `QMGR` angegeben wird.

public int CommandEvent {get;}
Gibt an, ob Befehlsereignisse generiert werden sollen.

public string CommandInputQueueName {get;}
Gibt den Namen der Warteschlange für Befehlseingaben zurück, die im Warteschlangenmanager definiert ist. Anwendungen können Befehle an diese Warteschlange senden, wenn sie über die entsprechende Berechtigung verfügen.

public int CommandLevel {get;}
Gibt die Funktionsebene des Warteschlangenmanagers an. Die Funktionsgruppe, die einer bestimmten Funktionsebene entspricht, hängt von der Plattform ab. Auf einer bestimmten Plattform können Sie jeden Warteschlangenmanager verlässlich verwenden, der die Funktionen auf der niedrigsten Funktionsebene unterstützt, die allen Warteschlangenmanagern gemeinsam ist.

public int CommandLevel {get;}
Gibt an, ob der Befehlsserver automatisch startet, wenn der Warteschlangenmanager startet.

public string DNSGroup {get;}
Der Name der Gruppe, zu der das TCP-Empfangsprogramm, das für die Behandlung eingehender Übertragungen für die Gruppe mit gemeinsamer Warteschlange zuständig ist, gehören sollte. Das Programm stellt eine Verbindung zu dieser Gruppe her, wenn es die WLM/DNS-Unterstützung (WLM/DNS = Workload Manager for Dynamic Domain Name Services) verwendet.

public int DNSWLM {get;}
Gibt an, ob das TCP-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange handhabt, beim Workload Manager für das Dynamic Domain Name System (DDNS) registriert werden muss.

public int IPAddressVersion {get;}
Gibt an, welches IP-Protokoll (IPv4 oder IPv6) für eine Kanalverbindung verwendet werden soll.

public boolean IsConnected {get;}
Gibt den Wert von `isConnected` zurück.

Wenn der Wert hierfür "True" (Wahr) lautet, wurde eine Verbindung zum Warteschlangenmanager hergestellt, von der keine Unterbrechung bekannt ist. Alle Aufrufe für "IsConnected" unternehmen keinen aktiven Versuch, den Warteschlangenmanager zu erreichen, daher besteht die Möglichkeit, dass die physische Verbindung unterbrochen ist, "IsConnected" jedoch weiterhin als Wert "True" zurückgibt. Der Status von "IsConnected" wird nur aktualisiert, wenn im Warteschlangenmanager eine Aktivität durchgeführt wird, beispielsweise eine Nachricht eingereiht oder abgerufen wird.

Wenn der Wert hierfür "False" (Falsch) lautet, wurde keine Verbindung zum Warteschlangenmanager hergestellt oder die Verbindung wurde unterbrochen.

public int KeepAlive {get;}
Gibt an, ob mithilfe der TCP-KEEPALIVE-Funktion überprüft werden soll, ob die andere Seite der Verbindung noch verfügbar ist. Wenn sie nicht mehr zur Verfügung steht, wird der Kanal geschlossen.

public int ListenerTimer {get;}
Das Zeitintervall (in Sekunden) zwischen den Versuchen von WebSphere MQ, das Empfangsprogramm nach einem APPC- oder TCP/IP-Ausfall erneut zu starten.

public int LoggerEvent {get;}
Gibt an, ob Protokollierungsereignisse generiert werden.

public string LU62ARMSuffix {get;}
Das Suffix des APPCPM-Elements von SYS1.PARMLIB. Dieses Suffix nominiert die LUADD für diesen Kanalinitiator. Wenn der Automatic Restart Manager (ARM) den Kanalinitiator neu startet, wird der z/OS-Befehl "SET APPC=xx" ausgegeben.

public string LUGroupName {get; z/os}
Der generische LU-Name, der vom LU 6.2-Empfangsprogramm verwendet werden soll, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange handhabt.

public string LUName {get;}
Der Name der LU, die für abgehende LU 6.2-Übertragungen verwendet werden soll.

public int MaximumActiveChannels {get;}
Die Anzahl an Kanälen, die maximal gleichzeitig aktiv sein können.

public int MaximumCurrentChannels {get;}
Die maximale Anzahl der Kanäle, die gleichzeitig aktiv sein können (einschließlich Serververbindungskanälen mit verbundenen Clients).

public int MaximumLU62Channels {get;}
Die maximale Anzahl an Kanälen, die gleichzeitig aktiv sein können, oder an Clients, die miteinander verbunden werden können und die das LU 6.2-Übertragungsprotokoll verwenden.

public int MaximumMessageLength {get;}
Gibt die maximale Nachrichtenlänge (in Bytes) zurück, die vom Warteschlangenmanager verarbeitet werden kann. Keine Warteschlange kann mit einer maximalen Nachrichtenlänge definiert werden, die größer ist als `MaximumMessageLength`.

public int MaximumPriority {get;}
Gibt die höchste Nachrichtenpriorität zurück, die vom Warteschlangenmanager unterstützt wird. Die Prioritäten reichen von null (niedrigste Priorität) bis zu diesem Wert. Löst `MQException` aus, wenn Sie diese Methode aufrufen, nachdem die Verbindung zum Warteschlangenmanager unterbrochen wurde.

public int MaximumTCPChannels {get;}
Die maximale Anzahl an Kanälen, die gleichzeitig aktiv sein können, oder an Clients, die miteinander verbunden werden können und die das TCP/IP-Übertragungsprotokoll verwenden.

public int MQIAccounting {get;}
Steuert die Erfassung von Abrechnungsinformationen für MQI-Daten.

public int MQIStatistics {get;}
Steuert die Erfassung von statistischen Überwachungsdaten für den Warteschlangenmanager.

public int OutboundPortMax {get;}
Der höchste Wert des Portnummernbereichs, der bei der Bindung abgehender Kanäle verwendet werden soll.

public int OutboundPortMin {get;}
Der niedrigste Wert des Portnummernbereichs, der bei der Bindung abgehender Kanäle verwendet werden soll.

public int QueueAccounting {get;}
Gibt an, ob Abrechnungsdaten der Klasse 3 (Abrechnung auf Thread- und Warteschlangenebene) für alle Warteschlangen verwendet werden sollen.

public int QueueMonitoring {get;}
Steuert die Erfassung von Onlineüberwachungsdaten für Warteschlangen.

public int QueueStatistics {get;}
Steuert die Erfassung von statistischen Daten für Warteschlangen.

public int ReceiveTimeout {get;}
Gibt an, wie lange ein TCP/IP-Kanal auf den Eingang von Daten (inklusive Überwachungssignalen) von der Partnerseite wartet, bevor er wieder in den inaktiven Status übergeht.

public int ReceiveTimeoutMin {get;}
Gibt an, wie lange ein TCP/IP-Kanal mindestens auf den Eingang von Daten (inklusive Überwachungssignalen) von der Partnerseite wartet, bevor er wieder in einen inaktiven Status übergeht.

public int ReceiveTimeoutType {get;}
Das Qualifikationsmerkmal, das für den Wert in `ReceiveTimeout` angewendet werden soll.

public int SharedQueueQueueManagerName {get;}
Gibt an, wie Nachrichten einer gemeinsam genutzten Warteschlange zugestellt werden sollen. Wenn die PUT-Operation einen anderen Warteschlangenmanager aus derselben Gruppe mit gemeinsamer Warteschlange als Zielwarteschlangenmanager angigt, wird die Nachricht auf zwei Wegen zugestellt:

MQC.MQSQQM_USE
Nachrichten werden dem Objektwarteschlangenmanager zugestellt, bevor sie in die gemeinsam genutzte Warteschlange eingereiht werden.

MQCMQSQQM_IGNORE
Nachrichten werden direkt in die gemeinsam genutzte Warteschlange eingereiht.

public int SSLEvent {get;}
Gibt an, ob SSL-Ereignisse generiert werden.

public int SSLFips {get;}
Gibt an, ob nur FIPS-zertifizierte Algorithmen verwendet werden sollen, wenn die Verschlüsselung in WebSphere MQ anstatt mit der Verschlüsselungshardware durchgeführt wird.

public int SSLKeyResetCount {get;}
Die Anzahl an unverschlüsselten Bytes, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet bzw. empfangen werden.

public int ClusterSenderStatistics {get;}
Gibt das Intervall (in Minuten) zwischen aufeinander folgenden Erfassungen von statistischen Daten an.

public int SyncpointAvailability {get;}
Gibt an, ob der Warteschlangenmanager Arbeitseinheiten und Synchronisationspunkte mit den Methoden `MQQueue.get` und `MQQueue.put` unterstützt.

public string TCPName {get;}
Der Name des einzigen bzw. standardmäßigen TCP/IP-Systems (je nach Wert für TCPStackType), das verwendet werden soll.

public int TCPStackType {get;}
Gibt an, ob der Kanalinitiator nur den TCP/IP-Adressraum verwendet, der in TCPName angegeben wurde. Alternativ kann der Kanalinitiator eine Bindung an eine beliebige TCP/IP-Adresse herstellen.

public int TraceRouteRecording {get;}
Steuert die Aufzeichnung von Informationen über die Tracefunktion für Routes.

Methoden

public MQProcess AccessProcess(string *processName*, int *openOptions*);
public MQProcess AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Löst `MQException` aus.

Greifen Sie auf einen WebSphere MQ-Prozess in diesem Warteschlangenmanager zu, um die Prozessattribute abzufragen.

processName
Der Name des zu öffnenden Prozesses.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs des Prozesses. Folgende gültige Optionen können hinzugefügt oder mit einem bitweisen ODER kombiniert werden:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
Der Name des Warteschlangenmanagers, auf dem der Prozess definiert ist. Sie können einen leeren Warteschlangenmanagernamen oder einen Nullwert für den Namen angeben, wenn der Warteschlangenmanager mit dem übereinstimmt, auf den der Prozess zugreift.

alternateUserId

Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY im Parameter *openOptions* angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für die Aktion überprüft wird. Wenn MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

Die Standardbenutzerberechtigung wird für die Verbindung zum Warteschlangenmanager verwendet, wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist.

public MQQueue AccessQueue(string *queueName*, int *openOptions*);
public MQQueue AccessQueue(string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Löst MQException aus.

Greift auf eine Warteschlange in diesem Warteschlangenmanager zu.

Sie können Nachrichten abrufen, durchsuchen oder einreihen und die Attribute der Warteschlange abfragen oder festlegen. Wenn die benannte Warteschlange eine Modellwarteschlange ist, wird eine dynamische lokale Warteschlange erstellt. Fragen Sie das name-Attribut des resultierenden MQQueue-Objekts ab, um den Namen der dynamischen Warteschlange zu ermitteln.

queueName
Name der zu öffnenden Warteschlange.

openOptions
Optionen zur Steuerung des Öffnungsvorgangs der Warteschlange.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Die Prüfung wird mit der angegebenen Benutzer-ID durchgeführt.

MQC.MQOO_BIND_AS_QDEF
Standardmäßige Bindung für Warteschlange verwenden.

MQC.MQOO_BIND_NOT_FIXED
Keine Bindung an ein bestimmtes Ziel.

MQC.MQOO_BIND_ON_OPEN
Bindung der Kennung an das Ziel, wenn die Warteschlange geöffnet wird.

MQC.MQOO_BROWSE
Zum Durchsuchen der Nachricht öffnen.

MQC.MQOO_FAIL_IF_QUIESCING
Fehler, wenn der Warteschlangenmanager stillgelegt wird.

MQC.MQOO_INPUT_AS_Q_DEF
Öffnen zum Empfangen von Nachrichten unter Verwendung der von der Warteschlange definierten Standardwerte.

MQC.MQOO_INPUT_SHARED
Öffnen zum Empfangen von Nachrichten mit gemeinsamem Zugriff.

MQC.MQOO_INPUT_EXCLUSIVE
Öffnen zum Empfangen von Nachrichten mit exklusivem Zugriff.

MQC.MQOO_INQUIRE
Öffnen für die Abfrage (erforderlich, wenn Sie Eigenschaften abfragen möchten).

MQC.MQOO_OUTPUT
Öffnen zum Einreihen von Nachrichten.

MQC.MQOO_PASS_ALL_CONTEXT
Weitergabe des gesamten Kontexts erlaubt.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Weitergabe des Identitätskontexts erlaubt.

MQC.MQOO_SAVE_ALL_CONTEXT
Speichern des Kontexts beim Abrufen einer Nachricht.

MQC.MQOO_SET
Öffnen zum Festlegen von Attributen (erforderlich, wenn Sie Eigenschaften festlegen möchten).

MQC.MQOO_SET_ALL_CONTEXT
Zulassen, dass der gesamte Kontext festgelegt wird.

MQC.MQOO_SET_IDENTITY_CONTEXT
Zulassen, dass der Identitätskontext festgelegt wird.

queueManagerName
Name des Warteschlangenmanagers, für den die Warteschlange definiert ist. Mit einem Namen, der völlig leer ist oder ausschließlich Nullwerte enthält, wird der Warteschlangenmanager angegeben, mit dem das MQQueueManager-Objekt verbunden ist.

dynamicQueueName
dynamicQueueName wird ignoriert, falls nicht mit queueName der Name einer Modellwarteschlange angegeben ist. In diesem Fall gibt *dynamicQueueName* den Namen der zu erstellenden dynamischen Warteschlange an. Ein leerer Name oder Nullname ist nicht gültig, wenn queueName den Namen einer Modellwarteschlange angibt. Wenn das letzte belegte Zeichen im Namen ein Stern (*) ist, ersetzt der Warteschlangenmanager den Stern durch eine Zeichenfolge. Mit den Zeichen wird sichergestellt, dass der für die Warteschlange generierte Name in diesem Warteschlangenmanager eindeutig ist.

alternateUserId
Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY im Parameter openOptions angegeben ist, gibt *alternateUserId* die alternative Benutzer-ID an, mit der die Berechtigung für den Öffnungsvorgang überprüft wird. Wenn MQC.MQOO_ALTERNATE_USER_AUTHORITY nicht angegeben ist, darf *alternateUserId* leer sein oder einen Nullwert annehmen.

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*);

Greift auf ein Thema in diesem Warteschlangenmanager zu.

MQTopic-Objekte stehen in einer engen Beziehung zu Verwaltungsthemenobjekten, die manchmal auch "Themenobjekte" genannt werden. Bei der Eingabe verweist `topicObject` auf ein Verwaltungsthemenobjekt. Der `MQTopic`-Konstruktor ruft eine Themenzeichenfolge vom Themenobjekt ab und kombiniert diese mit `topicName`, um einen Themennamen zu erstellen. Einer der beiden Werte (`topicObject` oder `topicName`) oder beide können null sein. Der Themenname wird mit der Themenstruktur abgeglichen, und der Name des am meisten übereinstimmenden Verwaltungsthemenobjekts wird in `topicObject` zurückgegeben.

Die dem `MQTopic`-Objekt zugeordneten Themen sind das Ergebnis der Kombination zweier Themenzeichenfolgen. Die erste Themenzeichenfolge wird mit dem Verwaltungsthemenobjekt definiert, das durch *topicObject* angegeben wird. Die zweite Themenzeichenfolge ist *topicString*. Die sich ergebende, dem `MQTopic`-Objekt zugeordnete Themenzeichenfolge kann durch das Einschließen von Platzhalterzeichen mehrere Themen angeben.

Abhängig davon, ob das Thema für die Veröffentlichung oder für die Subskription geöffnet ist, können Sie mithilfe der `MQTopic.Put`-Methoden die Veröffentlichung in Themen vornehmen oder mithilfe der `MQTopic.Get`-Methoden Veröffentlichungen in Themen empfangen. Wenn Sie die Veröffentlichung und die Subskription im selben Thema vornehmen möchten, müssen Sie zweimal auf das Thema zugreifen, einmal für die Veröffentlichung und einmal für die Subskription.

Wenn Sie ein `MQTopic`-Objekt für die Subskription erstellen, ohne ein `MQDestination`-Objekt bereitzustellen, wird davon ausgegangen, dass eine verwaltete Subskription vorliegt. Wenn Sie eine Warteschlange als `MQDestination`-Objekt übergeben, wird davon ausgegangen, dass eine nicht verwaltete Subskription vorliegt. Sie müssen sicherstellen, dass die Subskriptionsoptionen, die Sie festlegen, mit dem Status der Subskription als verwaltete oder nicht verwaltete Subskription übereinstimmen.

destination
destination ist eine `MQQueue`-Instanz. Wenn Sie *destination* bereitstellen, wird `MQTopic` als nicht verwaltete Subskription geöffnet. Veröffentlichungen zum Thema werden an die Warteschlange zugestellt, auf die als *destination*zugegriffen wird.

topicName
Eine Themenzeichenfolge, die den zweiten Teil eines Themennamens darstellt. *topicName* ist mit der im *topicObject*-Verwaltungsthemenobjekt definierten Themenzeichenfolge verkettet. Sie können *topicName* auf null setzen. In diesem Fall wird der Themenname durch die Themenzeichenfolge in *topicObject* definiert.

topicObject
Bei der Eingabe ist *topicObject* der Name des Themenobjekts, das die Themenzeichenfolge enthält, die den ersten Teil des Themennamens bildet. Die Themenzeichenfolge in *topicObject*

ist mit *topicName* verkettet. Die Regeln für die Erstellung von Themenzeichenfolgen sind im Abschnitt Themenzeichenfolgen kombinieren definiert.

Bei der Ausgabe enthält *topicObject* den Namen des Verwaltungsthemenobjekts, das die größte Übereinstimmung in der Themenstruktur mit dem durch den Themennamen angegebenen Thema aufweist.

openAs
Greift auf das Thema für eine Veröffentlichung oder Subskription zu. Der Parameter kann nur eine dieser Optionen enthalten:

- `MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION`
- `MQC.MQTOPIC_OPEN_AS_PUBLICATION`

options

Kombinieren Sie die Optionen, mit denen das Öffnen des Themas entweder für die Veröffentlichung oder für die Subskription gesteuert wird. Mit den `MQC.MQSO_*`-Konstanten können Sie auf ein Thema für die Subskription zugreifen, mit den `MQC.MQOO_*`-Konstanten können Sie auf ein Thema für die Veröffentlichung zugreifen.

Wenn mehrere Optionen erforderlich sind, addieren Sie die Werte zusammen oder kombinieren Sie die Optionswerte mit dem bitweisen Operator `OR` .

alternateUserId
Geben Sie die alternative Benutzer-ID an, die zur Überprüfung der erforderlichen Berechtigung zum Beenden des Vorgangs erforderlich ist. Sie müssen *alternateUserId* angeben, wenn entweder `MQC.MQOO_ALTERNATE_USER_AUTHORITY` oder `MQC.MQSO_ALTERNATE_USER_AUTHORITY` im Optionsparameter festgelegt ist.

subscriptionName
subscriptionName ist erforderlich, wenn die Option `MQC.MQSO_DURABLE` oder `MQC.MQSO_ALTER` bereitgestellt wird. In beiden Fällen ist `MQTopic` implizit für die Subskription geöffnet. Eine Ausnahme wird ausgelöst, wenn `MQC.MQSO_DURABLE` festgelegt ist und die Subskription vorhanden ist oder wenn `MQC.MQSO_ALTER` festgelegt ist und die Subskription nicht vorhanden ist.

properties
Legen Sie eine der besonderen Subskriptionseigenschaften fest, die mithilfe einer Hashtabelle aufgelistet wurden. In der Hashtabelle angegebene Einträge werden mit Ausgabewerten aktualisiert. Einträge werden nicht zur Hashtabelle hinzugefügt, um Berichte für Ausgabewerte zu erstellen.

- `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();`

Löst `MQException` aus.

Gibt ein `MQAsyncStatus`-Objekt zurück, das die asynchrone Aktivität für die Verbindung des Warteschlangenmanagers darstellt.

`public void Backout();`

Löst `MQException` aus.

Setzt alle Nachrichten zurück, die innerhalb des Synchronisationspunktes seit dem letzten Synchronisationspunkt gelesen oder geschrieben wurden.

Nachrichten, die mit dem Flag MQC.MQPMO_SYNCPOINT geschrieben wurden, werden aus Warteschlangen entfernt. Nachrichten, die mit dem Flag MQC.MQGMO_SYNCPOINT gelesen werden, werden in den Warteschlangen wiederhergestellt, aus denen sie stammen. Wenn es sich um persistente Nachrichten handelt, werden die Änderungen protokolliert.

Bei wiederverbindbaren Clients wird der Ursachencode MQRC_NONE an einen Client zurückgegeben, nachdem die Verbindung erfolgreich wiederhergestellt wurde.

public void Begin();

Löst MQException aus.

Begin wird nur im Serverbindungsmodus unterstützt. Damit wird eine globale Arbeitseinheit gestartet.

public void Commit();

Löst MQException aus.

Schreibt alle Nachrichten fest, die innerhalb des Synchronisationspunktes seit dem letzten Synchronisationspunkt gelesen oder geschrieben wurden.

Nachrichten, die mit dem Flag MQC.MQPMO_SYNCPOINT geschrieben werden, werden anderen Anwendungen zur Verfügung gestellt. Mit dem Flag MQC.MQGMO_SYNCPOINT abgerufene Nachrichten werden gelöscht. Wenn es sich um persistente Nachrichten handelt, werden die Änderungen protokolliert.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- MQRC_CALL_INTERRUPTED, wenn die Verbindung während der Ausführung des Festschreibungsaufrufs unterbrochen wird.
- MQRC_BACKED_OUT, wenn der Commit-Aufruf nach der Verbindungswiederherstellung ausgegeben wird.

Disconnect();

Löst MQException aus.

Schließt die Verbindung zum Warteschlangenmanager. Alle Objekte, auf die in diesem Warteschlangenmanager zugegriffen wird, stehen für diese Anwendung nicht mehr zur Verfügung. Um erneut auf die Objekte zuzugreifen, erstellen Sie ein MQQueueManager-Objekt.

Generell wird jede Arbeit, die als Teil einer Arbeitseinheit durchgeführt wird, festgeschrieben. Wenn die Arbeitseinheit jedoch mit .NET verwaltet wird, wird möglicherweise ein Rollback für die Arbeitseinheit durchgeführt.

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*);

Löst MQException aus.

Legt eine einzelne Nachricht in einer Warteschlange oder in einem Thema ab, ohne zuerst ein MQQueue- oder MQTopic-Objekt zu erstellen.

queueName
Der Name der Warteschlange, in der die Nachricht abgelegt werden soll.

destinationName
Der Name eines Zielobjekts. Je nach Wert von *type* handelt es sich entweder um eine Warteschlange oder um ein Thema.

type
Der Typ des Zielobjekts. Sie dürfen die Optionen nicht kombinieren.

MQC.MQOT_Q
Warteschlange

MQC.MQOT_TOPIC
Thema

queueManagerName

Der Name oder Aliasname des Warteschlangenmanagers, für den die Warteschlange definiert ist. Bei Angabe des Typs `MQC.MQOT_TOPIC` wird der Parameter ignoriert.

Wenn es sich um eine Modellwarteschlange handelt und der aufgelöste Warteschlangenmanagername nicht diesem Warteschlangenmanager entspricht, wird die Ausnahme `MQException` ausgelöst.

topicString

topicString wird mit dem Themennamen im Themenobjekt *destinationName* kombiniert.

topicString wird ignoriert, wenn *destinationName* eine Warteschlange ist.

message
Die zu sendende Nachricht. Die Nachricht ist ein Ein-/Ausgabeobjekt.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- `MQRC_CALL_INTERRUPTED`, wenn die Verbindung unterbrochen wird, während ein Put-Aufruf für eine persistente Nachricht ausgeführt wird.
- `MQRC_NONE`, wenn die Verbindung erfolgreich hergestellt wurde, während ein PUT-Aufruf für eine nicht persistente Nachricht durchgeführt wurde (siehe Anwendungswiederherstellung).

putMessageOptions

Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn Sie *putMessageOptions* nicht angeben, wird eine Standardinstanz von *putMessageOptions* erstellt. *putMessageOptions* ist ein Ein-/Ausgabeobjekt.

Wenn Sie die Option `MQPMO_LOGICAL_ORDER` in einem wiederverbindbaren Client verwenden, wird der Ursachencode `MQRC_RECONNECT_INCOMPATIBLE` zurückgegeben.

alternateUserId

Gibt beim Einreihen der Nachricht in eine Warteschlange eine alternative Benutzer-ID zur Überprüfung der Berechtigung an.

Sie können *alternateUserId* weglassen, wenn Sie `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nicht in *putMessageOptions* festlegen. Wenn Sie `MQC.MQOO_ALTERNATE_USER_AUTHORITY` festlegen, müssen Sie auch *alternateUserId* festlegen. *alternateUserId* hat keine Auswirkung, wenn Sie nicht auch `MQC.MQOO_ALTERNATE_USER_AUTHORITY` festlegen.

Konstruktoren

```
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 connNa-
me);
public MQQueueManager(string queueManagerName, System.Collections.Hashtable
properties);
```

Löst `MQException` aus.

Erstellt eine Verbindung zu einem Warteschlangenmanager. Wählen Sie aus, ob Sie eine Client- oder eine Serververbindung erstellen möchten.

Sie müssen über die Berechtigung zum Abfragen (`inq`) im Warteschlangenmanager verfügen, wenn Sie versuchen, eine Verbindung zum Warteschlangenmanager herzustellen. Ohne die Berechtigung zum Abfragen schlägt der Verbindungsversuch fehl.

Eine Clientverbindung wird erstellt, wenn eine der folgenden Bedingungen erfüllt ist:

1. *channel* oder *connName* werden im Konstruktor angegeben.
2. *HostName*, *Port* oder *Channel* sind in *properties* angegeben.
3. *MQEnvironment.HostName*, *MQEnvironment.Port* oder *MQEnvironment.Channel* sind angegeben.

Die Werte der Verbindungseigenschaft werden standardmäßig in der angezeigten Reihenfolge dargestellt. *channel* und *connName* im Konstruktor haben Vorrang vor den Eigenschaftswerten im Konstruktor. Die Eigenschaftswerte des Konstruktors haben Vorrang vor den `MQEnvironment`-Eigenschaften.

Der Hostname, der Kanalname und der Port sind in der `MQEnvironment`-Klasse definiert.

queueManagerName
Name des Warteschlangenmanagers oder der Gruppe von Warteschlangenmanagern, zu dem bzw. der eine Verbindung hergestellt werden soll.

Übergehen Sie den Parameter, behalten Sie den Wert null bei oder lassen Sie ihn leer, um eine Standardauswahl für den Warteschlangenmanager zu treffen. Die Standardverbindung für den Warteschlangenmanager auf einem Server wird zum standardmäßigen Warteschlangenmanager auf dem Server hergestellt. Die Standardverbindung für den Warteschlangenmanager bei einer Clientverbindung wird zu dem Warteschlangenmanager hergestellt, mit dem das Empfangsprogramm verbunden ist.

options
Geben Sie `MQCNO`-Verbindungsoptionen an. Die Werte müssen für den Typ der gewünschten Verbindung gültig sein. Beispiel: Wenn Sie die folgenden Serververbindungseigenschaften für eine Clientverbindung angeben, wird `MQException` ausgelöst.

- `MQC.MQCNO_FASTPATH_BINDING`
- `MQC.MQCNO_STANDARD_BINDING`

properties

Der Eigenschaftenparameter übernimmt eine Reihe von Schlüssel/Wert-Paaren, mit denen die von `MQEnvironment` festgelegten Eigenschaften überschrieben werden; siehe das Beispiel unter „MQEnvironment-Eigenschaften überschreiben” auf Seite 1332. Folgende Eigenschaften können überschrieben werden:

- `MQC.CONNECT_OPTIONS_PROPERTY`
- `MQC.CONNECTION_NAME_PROPERTY`
- `MQC.ENCRYPTION_POLICY_SUITE_B`

- MQC.HOST_NAME_PROPERTY
- MQC.PORT_PROPERTY
- MQC.CHANNEL_PROPERTY
- MQC.SSL_CIPHER_SPEC_PROPERTY
- MQC.SSL_PEER_NAME_PROPERTY
- MQC.SSL_CERT_STORE_PROPERTY
- MQC.SSL_CRYPTO_HARDWARE_PROPERTY
- MQC.SECURITY_EXIT_PROPERTY
- MQC.SECURITY_USERDATA_PROPERTY
- MQC.SEND_EXIT_PROPERTY
- MQC.SEND_USERDATA_PROPERTY
- MQC.RECEIVE_EXIT_PROPERTY
- MQC.RECEIVE_USERDATA_PROPERTY
- MQC.USER_ID_PROPERTY
- MQC.PASSWORD_PROPERTY
- MQC.MQAIR_ARRAY
- MQC.KEY_RESET_COUNT
- MQC.FIPS_REQUIRED
- MQC.HDR_CMP_LIST
- MQC.MSG_CMP_LIST
- MQC.TRANSPORT_PROPERTY

channel
Name eines Serververbindungskanals.

connName
Verbindungsname im Format *HostName* (*Port*).

Sie können eine Liste mit *Hostnamen* und *Ports* als Argument für den Konstruktor `MQQueueManager(String queueManagerName, Hashtable properties)` mit CONNECTION_NAME_PROPERTY angeben.

Beispiel:

```
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);
```

Wenn ein Verbindungsversuch unternommen wird, wird die Liste der Verbindungsnamen der Reihe nach verarbeitet. Wenn der Verbindungsversuch zum ersten Hostnamen und Port fehlschlägt, wird versucht, eine Verbindung zum zweiten Attributpaar herzustellen. Der Client wiederholt diesen Prozess, bis entweder eine erfolgreiche Verbindung hergestellt wurde oder das Ende der Liste erreicht ist. Wenn das Ende der Liste erreicht ist, werden ein entsprechender Beendigungs- und Ursachencode an die Clientanwendung zurückgegeben.

Wenn für den Verbindungsnamen keine Portnummer angegeben ist, wird der Standardport (unter `mqclient.ini` konfiguriert) verwendet.

Verbindungsliste festlegen

Sie können die Verbindungsliste mit den folgenden Methoden festlegen, wenn die Optionen für die automatische Verbindungswiederholung des Clients festgelegt sind:

Verbindungsliste über MQSERVER festlegen

Sie können die Verbindungsliste über die Eingabeaufforderung festlegen.

Nehmen Sie an der Eingabeaufforderung folgende Einstellung vor:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
For Example:
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Wenn Sie die Verbindung in MQSERVER festgelegt haben, legen Sie sie nicht in der Anwendung fest.

Wenn Sie die Verbindungsliste in der Anwendung festlegen, überschreibt die Anwendung alle Einstellungen in der MQSERVER-Umgebungsvariablen.

Verbindungsliste über die Anwendung festlegen

Sie können die Verbindungsliste in der Anwendung festlegen, indem Sie die Eigenschaften für den Hostnamen und Port angeben.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Verbindungsliste über `app.config` festlegen

`App.config` ist eine XML-Datei, in der Sie die Schlüssel/Wert-Paare angeben.

Nehmen Sie in der Verbindungsliste folgende Einstellungen vor:

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
<app.Settings>
```

Beispiel:

```
<app.Settings>
<add key>="Connection1" value="fred.mq.com(2966)"/>
<add key>="Connection2" value="alex.mq.com(6533)"/>
<app.Settings>
```

Sie können die Verbindungsliste direkt in der `app.config` -Datei ändern.

Verbindungsliste über `MQEnvironment` festlegen

Um die Verbindungsliste über `MQEnvironment` festzulegen, verwenden Sie die Eigenschaft *ConnectionName*.

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),"alex.mq.com(5211);
```

Die Eigenschaft *ConnectionName* überschreibt die unter `MQEnvironment` festgelegten Eigenschaften für den Hostnamen und Port.

Clientverbindung erstellen

Im folgenden Beispiel wird gezeigt, wie Sie eine Clientverbindung zu einem Warteschlangenmanager erstellen. Sie können eine Clientverbindung erstellen, indem Sie die `MQEnvironment`-Variablen festlegen, bevor Sie ein neues `MQQueueManager`-Objekt erstellen.

```
MQEnvironment.Hostname = "fred.mq.com";  // host to connect to
MQEnvironment.Port     = 1414;           // port to connect to
                                         //If not explicitly set,
                                         // defaults to 1414
                                         // (the default WebSphere MQ port)
MQEnvironment.Channel  = "channel.name"; // the case sensitive
                                         //  name of the
                                         // SVR CONN channel on
                                         // the queue manager
MQQueueManager qMgr    = new MQQueueManager("MYQM");
```

Abbildung 43. Clientverbindung

MQEnvironment-Eigenschaften überschreiben

Im folgenden Beispiel wird gezeigt, wie Sie einen Warteschlangenmanager mit der zugehörigen Benutzer-ID und dem zugehörigen Kennwort erstellen, die in einer Hashtabelle definiert sind.

```
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);
}
```

Abbildung 44. MQEnvironment-Eigenschaften überschreiben

Wiederherstellbare Verbindung erstellen

Im folgenden Beispiel wird gezeigt, wie Sie die Verbindung eines Client zu einem Warteschlangenmanager automatisch wiederherstellen.

```
Hashtable properties = new Hashtable(); // The queue manager name and the
                                        // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT);//Options through which
                                                    // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY,"fred.mq.com(4789),nick.mq.com(4790)");// The list
of
                                                  // queue managers through which reconnect
happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

Abbildung 45. Automatisch die Verbindung zwischen Client und Warteschlangenmanager wiederherstellen

.NET-Klasse MQSubscription

Mit MQSubscription können Sie anfordern, dass ständige Veröffentlichungen an den Subskribenten gesendet werden. MQSubscription ist eine Eigenschaft eines MQTopic-Objekts, das für die Subskription geöffnet ist.

Klasse

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQSubscription
```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- „Eigenschaften” auf Seite 1333
- „Methoden” auf Seite 1333
- „Konstruktoren” auf Seite 1333

Eigenschaften

Greifen Sie mithilfe der MQManagedObject-Klasse auf die Subskriptionseigenschaften zu; weitere Informationen finden Sie unter „Eigenschaften” auf Seite 1290.

Methoden

Greifen Sie auf die Subskriptionsmethoden Inquire, Set und Get zu. Verwenden Sie hierfür die MQManagedObject-Klasse; weitere Informationen finden Sie unter „Methoden” auf Seite 1291.

public int RequestPublicationUpdate(int *options*);

Löst MQException aus.

Fordern Sie eine aktualisierte Veröffentlichung für das aktuelle Thema an. Wenn der Warteschlangenmanager über ständige Veröffentlichungen für das Thema verfügt, werden diese an den Subskribenten gesendet.

Bevor Sie RequestPublicationUpdate aufrufen, öffnen Sie ein Thema für die Subskription, um ein MQSubscription-Objekt zu erhalten.

Üblicherweise öffnen Sie die Subskription mit der Option MQC.MQSO_PUBLICATIONS_ON_REQUEST. Wenn in der Themenzeichenfolge keine Platzhalterzeichen vorhanden sind, wird als Ergebnis dieses Aufrufs nur eine Veröffentlichung gesendet. Wenn die Themenzeichenfolge Platzhalterzeichen enthält, werden möglicherweise viele Veröffentlichungen gesendet. Die Methode gibt die Anzahl der ständigen Veröffentlichungen zurück, die an die Subskriptionswarteschlange gesendet werden. Es gibt keine Garantie dafür, dass genau so viele Veröffentlichungen empfangen werden, insbesondere wenn nicht persistente Nachrichten vorliegen.

options

MQC.MQSRO_FAIL_IF_QUIESCING
Die Methode schlägt fehl, wenn sich der Warteschlangenmanager im Stilllegungsmodus befindet. Unter z/OS wird bei einer CICS- oder IMS-Anwendung mit MQC.MQSRO_FAIL_IF_QUIESCING das Fehlschlagen der Methode auch dann erzwungen, wenn sich die Verbindung im Stilllegungsmodus befindet.

MQC.MQSRO_NONE
Es werden keine Optionen angegeben.

Konstruktoren

Kein öffentlicher Konstruktor.

Ein MQSubscription-Objekt wird in der SubscriptionReference-Eigenschaft eines MQTopic-Objekts zurückgegeben, das für die Subskription geöffnet ist.

Rufen Sie die RequestPublicationUpdate-Methode auf. MQSubscription ist eine Unterklasse von MQManagedObject. Verwenden Sie die Referenz, um auf die Eigenschaften und Methoden von MQManagedObject zuzugreifen.

.NET-Klasse MQTopic

Mit MQTopic können Sie Nachrichten in einem Thema veröffentlichen oder subskribieren oder die Attribute eines Themas abfragen oder festlegen. Erstellen Sie ein MQTopic-Objekt für die Veröffentlichung oder Subskription mithilfe eines Konstruktors oder der Methode MQQueueManager.AccessTopic.

Klasse

```
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;
```


- „Eigenschaften” auf Seite 1334
- „Methoden” auf Seite 1334
- „Konstruktoren” auf Seite 1336

Eigenschaften

Test für den Fall, dass MQException beim Abrufen von Eigenschaften ausgelöst wird.

public Boolean IsDurable {get;}
Schreibgeschützte Eigenschaft, die True zurückgibt, wenn die Subskription permanent ist. Andernfalls wird False zurückgegeben. Wenn das Thema für die Veröffentlichung geöffnet wurde, wird die Eigenschaft ignoriert und immer False zurückgegeben.

public Boolean IsManaged {get;};
Schreibgeschützte Eigenschaft, die True zurückgibt, wenn die Subskription vom Warteschlangenmanager verwaltet wird. Andernfalls wird False zurückgegeben. Wenn das Thema für die Veröffentlichung geöffnet wurde, wird die Eigenschaft ignoriert und immer "False" zurückgegeben.

public Boolean IsSubscribed {get;};
Schreibgeschützte Eigenschaft, die True zurückgibt, wenn das Thema für die Subskription geöffnet wurde, und False, wenn das Thema für die Veröffentlichung geöffnet wurde.

public MQSubscription SubscriptionReference {get;};
Schreibgeschützte Eigenschaft, mit der ein MQSubscription-Objekt zurückgegeben wird, das einem für die Subskription geöffneten Themenobjekt zugeordnet ist. Die Referenz ist nur verfügbar, wenn Sie die Optionen zum Schließen ändern oder eine beliebige Objektmethode starten.

public MQDestination UnmanagedDestinationReference {get;};
Schreibgeschützte Eigenschaft, von der die MQQueue-Warteschlange zurückgegeben wird, die einer nicht verwalteten Subskription zugeordnet ist. Dabei handelt es sich um das Ziel, das beim Erstellen des Themenobjekts angegeben wurde. Die Eigenschaft gibt für sämtliche Themenobjekte, die für die Veröffentlichung oder mit einer verwalteten Subskription geöffnet sind, null zurück.

Methoden

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
Löst MQException aus.

Veröffentlicht eine Nachricht für das Thema.

Änderungen am MQMessage-Objekt, die nach Abschluss des PUT-Aufrufs vorgenommen werden, wirken sich nicht auf die tatsächliche Nachricht in der WebSphere MQ-Warteschlange oder im Veröffentlichungsthema aus.

Mit Put werden die MessageId- und CorrelationId-Eigenschaften des MQMessage-Objekts aktualisiert. Dabei werden keine Nachrichtendaten gelöscht. Weitere Put- oder Get-Aufrufe beziehen sich auf die aktualisierten Informationen im MQMessage-Objekt. Beispiel: Im folgenden Codeausschnitt enthält die erste Nachricht a und die zweite ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
```

```
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Ein MQMessage-Objekt, das die Nachrichtendeskriptordaten und die zu sendende Nachricht enthält. Der Nachrichtendeskriptor kann sich aufgrund dieser Methode ändern. Die Werte im Nachrichtendeskriptor sofort nach Abschluss dieser Methode sind die Werte, die in die Warteschlange eingereiht oder für das Thema veröffentlicht wurden.

Folgende Ursachencodes werden an einen Client zurückgegeben, dessen Verbindung wiederhergestellt werden kann:

- MQRC_CALL_INTERRUPTED, wenn die Verbindung unterbrochen wird, während ein PUT-Aufruf für eine persistente Nachricht durchgeführt wird und die Verbindungswiederholung erfolgreich ist.
- MQRC_NONE, wenn die Verbindung erfolgreich hergestellt wurde, während ein PUT-Aufruf für eine nicht persistente Nachricht durchgeführt wurde (siehe Anwendungswiederherstellung).

putMessageOptions
Optionen zum Steuern der Aktionen des Put-Vorgangs.

Wenn *putMessageOptions* nicht angegeben ist, wird die Standardinstanz von MQPutMessageOptons verwendet.

Wenn Sie die Option MQPMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

Anmerkung: Wenn Sie aus Gründen der Einfachheit und Leistungsoptimierung eine einzelne Nachricht in eine Warteschlange einreihen möchten, verwenden Sie das MQQueueManager.Put-Objekt. Hierfür sollten Sie über ein MQQueue-Objekt verfügen.

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);
Löst MQException aus.

Ruft eine Nachricht aus dem Thema ab.

Diese Methode verwendet eine Standardinstanz von MQGetMessageOptions für den Abruf. Die Nachrichtenoption lautet MQGMO_NOWAIT.

Wenn der Abrufvorgang fehlschlägt, bleibt das MQMessage-Objekt unverändert. Wenn der Vorgang erfolgreich ist, werden der Nachrichtendeskriptor und Teile der Nachrichtendaten von MQMessage durch den Nachrichtendeskriptor und die Nachrichtendaten aus der eingehenden Nachricht ersetzt.

Alle Aufrufe für WebSphere MQ von einem bestimmten MQQueueManager sind synchron. Wenn Sie daher einen Abruf mit Wartestatus durchführen, werden alle anderen Threads, die denselben MQQueueManager verwenden, blockiert und können keine weiteren WebSphere MQ-Aufrufe durchführen, bis der GET-Aufruf abgeschlossen ist. Wenn Sie mehrere Threads benötigen, um gleichzeitig auf WebSphere MQ zuzugreifen, muss jeder Thread sein eigenes MQQueueManager-Objekt erstellen.

message
Enthält der Nachrichtendeskriptor und die zurückgegebenen Nachrichtendaten. Einige Felder im Nachrichtendeskriptor sind Eingabeparameter. Es ist wichtig, sicherzustellen, dass die MessageId- und CorrelationId-Eingabeparameter als erforderlich festgelegt werden.

Ein Client, für den die Verbindung wiederhergestellt werden kann, gibt nach der erfolgreichen Verbindungswiederholung den Ursachencode MQRC_BACKED_OUT für Nachrichten zurück, die unter MQGM_SYNCPOINT empfangen wurden.

getMessageOptions
Optionen zum Steuern der Aktionen des Abrufs.

Die Verwendung von MQC.MQGMO_CONVERT führt möglicherweise zu einer Ausnahme mit dem Ursachencode MQC.MQRC_CONVERTED_STRING_TOO_BIG, wenn Sie eine Konvertierung von Co-

des mit Einzelbytezeichen in Codes mit Doppelbytezeichen durchführen. In diesem Fall wird die Nachricht ohne Konvertierung in den Zwischenspeicher kopiert.

Wenn *getMessageOptions* nicht angegeben wird, lautet die verwendete Nachrichtenoption MQGMO_NOWAIT.

Wenn Sie die Option MQGMO_LOGICAL_ORDER in einem wiederverbindbaren Client verwenden, wird der Ursachencode MQRC_RECONNECT_INCOMPATIBLE zurückgegeben.

MaxMsgSize
Die größte Nachricht, die dieses Nachrichtenobjekt erhalten darf. Wenn die Nachricht in der Warteschlange größer ist als dieser Wert, tritt eine der beiden folgenden Situationen ein:

- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag im MQGetMessageOptions-Objekt gesetzt ist, werden möglichst viele Nachrichtendaten in die Nachricht übernommen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_ACCEPTED ausgelöst.
- Wenn das MQGMO_ACCEPT_TRUNCATED_MSG-Flag nicht gesetzt wird, wird die Nachricht in der Warteschlange belassen. Dabei wird eine Ausnahme mit dem Beendigungscode MQCC_WARNING und dem Ursachencode MQRC_TRUNCATED_MSG_FAILED ausgelöst.

Wenn *MaxMsgSize* nicht angegeben ist, wird die gesamte Nachricht abgerufen.

Konstruktoren

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*);

Greift auf ein Thema im Warteschlangenmanager (*queueManager*) zu.

MQTopic-Objekte stehen in einer engen Beziehung zu Verwaltungsthemenobjekten, die manchmal auch "Themenobjekte" genannt werden. Bei der Eingabe verweist topicObject auf ein Verwaltungsthemenobjekt. Der MQTopic-Konstruktor ruft eine Themenzeichenfolge vom Themenobjekt ab und kombiniert diese mit topicName, um einen Themennamen zu erstellen. Einer der beiden Werte (topicObject oder topicName) oder beide können null sein. Der Themenname wird mit der Themenstruktur abgeglichen, und der Name des am meisten übereinstimmenden Verwaltungsthemenobjekts wird in topicObject zurückgegeben.

Die dem MQTopic-Objekt zugeordneten Themen sind das Ergebnis der Kombination zweier Themenzeichenfolgen. Die erste Themenzeichenfolge wird mit dem Verwaltungsthemenobjekt definiert, das durch *topicObject* angegeben wird. Die zweite Themenzeichenfolge ist *topicString*. Die sich ergebende, dem MQTopic-Objekt zugeordnete Themenzeichenfolge kann durch das Einschließen von Platzhalterzeichen mehrere Themen angeben.

Abhängig davon, ob das Thema für die Veröffentlichung oder für die Subskription geöffnet ist, können Sie mithilfe der MQTopic.Put-Methoden die Veröffentlichung in Themen vornehmen oder mithilfe der MQTopic.Get-Methoden Veröffentlichungen in Themen empfangen. Wenn Sie die Veröffentlichung und die Subskription im selben Thema vornehmen möchten, müssen Sie zweimal auf das Thema zugreifen, einmal für die Veröffentlichung und einmal für die Subskription.

Wenn Sie ein MQTopic-Objekt für die Subskription erstellen, ohne ein MQDestination-Objekt bereitzustellen, wird davon ausgegangen, dass eine verwaltete Subskription vorliegt. Wenn Sie eine Warteschlange als MQDestination-Objekt übergeben, wird davon ausgegangen, dass eine nicht verwaltete Subskription vorliegt. Sie müssen sicherstellen, dass die Subskriptionsoptionen, die Sie festlegen, mit dem Status der Subskription als verwaltete oder nicht verwaltete Subskription übereinstimmen.

queueManager
Warteschlangenmanager für den Zugriff auf ein Thema.

destination
destination ist eine MQQueue-Instanz. Wenn Sie *destination* bereitstellen, wird MQTopic als nicht verwaltete Subskription geöffnet. Veröffentlichungen zum Thema werden an die Warteschlange zugestellt, auf die als *destination*zugegriffen wird.

topicName
Eine Themenzeichenfolge, die den zweiten Teil eines Themennamens darstellt. *topicName* ist mit der im *topicObject*-Verwaltungsthemenobjekt definierten Themenzeichenfolge verkettet. Sie können *topicName* auf null setzen. In diesem Fall wird der Themenname durch die Themenzeichenfolge in *topicObject* definiert.

topicObject
Bei der Eingabe ist *topicObject* der Name des Themenobjekts, das die Themenzeichenfolge enthält, die den ersten Teil des Themennamens bildet. Die Themenzeichenfolge in *topicObject* ist mit *topicName* verkettet. Die Regeln für die Erstellung von Themenzeichenfolgen sind im Abschnitt Themenzeichenfolgen kombinieren definiert.

Bei der Ausgabe enthält *topicObject* den Namen des Verwaltungsthemenobjekts, das die größte Übereinstimmung in der Themenstruktur mit dem durch den Themennamen angegebenen Thema aufweist.

openAs
Greift auf das Thema für eine Veröffentlichung oder Subskription zu. Der Parameter kann nur eine dieser Optionen enthalten:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

Kombinieren Sie die Optionen, mit denen das Öffnen des Themas entweder für die Veröffentlichung oder für die Subskription gesteuert wird. Mit den MQC.MQSO_*-Konstanten können Sie auf ein Thema für die Subskription zugreifen, mit den MQC.MQOO_*-Konstanten können Sie auf ein Thema für die Veröffentlichung zugreifen.

Wenn mehrere Optionen erforderlich sind, addieren Sie die Werte zusammen oder kombinieren Sie die Optionswerte mit dem bitweisen Operator OR .

alternateUserId
Geben Sie die alternative Benutzer-ID an, die zur Überprüfung der erforderlichen Berechtigung zum Beenden des Vorgangs erforderlich ist. Sie müssen *alternateUserId* angeben, wenn entweder MQC.MQOO_ALTERNATE_USER_AUTHORITY oder MQC.MQSO_ALTERNATE_USER_AUTHORITY im Optionsparameter festgelegt ist.

subscriptionName
subscriptionName ist erforderlich, wenn die Option MQC.MQSO_DURABLE oder MQC.MQSO_ALTER bereitgestellt wird. In beiden Fällen ist MQTopic implizit für die Subskription geöffnet. Eine Ausnahme wird ausgelöst, wenn MQC.MQSO_DURABLE festgelegt ist und die Subskription vorhanden ist oder wenn MQC.MQSO_ALTER festgelegt ist und die Subskription nicht vorhanden ist.

properties
Legen Sie eine der besonderen Subskriptionseigenschaften fest, die mithilfe einer Hashtabelle aufgelistet wurden. In der Hashtabelle angegebene Einträge werden mit Ausgabewerten aktualisiert. Einträge werden nicht zur Hashtabelle hinzugefügt, um Berichte für Ausgabewerte zu erstellen.

- 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*);

Greift auf ein Thema in diesem Warteschlangenmanager zu.

MQTopic-Objekte stehen in einer engen Beziehung zu Verwaltungsthemenobjekten, die manchmal auch "Themenobjekte" genannt werden. Bei der Eingabe verweist topicObject auf ein Verwaltungsthemenobjekt. Der MQTopic-Konstruktor ruft eine Themenzeichenfolge vom Themenobjekt ab und kombiniert diese mit topicName, um einen Themennamen zu erstellen. Einer der beiden Werte (topicObject oder topicName) oder beide können null sein. Der Themenname wird mit der Themenstruktur abgeglichen, und der Name des am meisten übereinstimmenden Verwaltungsthemenobjekts wird in topicObject zurückgegeben.

Die dem MQTopic-Objekt zugeordneten Themen sind das Ergebnis der Kombination zweier Themenzeichenfolgen. Die erste Themenzeichenfolge wird mit dem Verwaltungsthemenobjekt definiert, das durch *topicObject* angegeben wird. Die zweite Themenzeichenfolge ist *topicString*. Die sich ergebende, dem MQTopic-Objekt zugeordnete Themenzeichenfolge kann durch das Einschließen von Platzhalterzeichen mehrere Themen angeben.

Abhängig davon, ob das Thema für die Veröffentlichung oder für die Subskription geöffnet ist, können Sie mithilfe der MQTopic.Put-Methoden die Veröffentlichung in Themen vornehmen oder mithilfe der MQTopic.Get-Methoden Veröffentlichungen in Themen empfangen. Wenn Sie die Veröffentlichung und die Subskription im selben Thema vornehmen möchten, müssen Sie zweimal auf das Thema zugreifen, einmal für die Veröffentlichung und einmal für die Subskription.

Wenn Sie ein MQTopic-Objekt für die Subskription erstellen, ohne ein MQDestination-Objekt bereitzustellen, wird davon ausgegangen, dass eine verwaltete Subskription vorliegt. Wenn Sie eine Warteschlange als MQDestination-Objekt übergeben, wird davon ausgegangen, dass eine nicht

verwaltete Subskription vorliegt. Sie müssen sicherstellen, dass die Subskriptionsoptionen, die Sie festlegen, mit dem Status der Subskription als verwaltete oder nicht verwaltete Subskription übereinstimmen.

destination
destination ist eine MQQueue-Instanz. Wenn Sie *destination* bereitstellen, wird MQTopic als nicht verwaltete Subskription geöffnet. Veröffentlichungen zum Thema werden an die Warteschlange zugestellt, auf die als *destination*zugegriffen wird.

topicName
Eine Themenzeichenfolge, die den zweiten Teil eines Themennamens darstellt. *topicName* ist mit der im *topicObject*-Verwaltungsthemenobjekt definierten Themenzeichenfolge verkettet. Sie können *topicName* auf null setzen. In diesem Fall wird der Themenname durch die Themenzeichenfolge in *topicObject* definiert.

topicObject
Bei der Eingabe ist *topicObject* der Name des Themenobjekts, das die Themenzeichenfolge enthält, die den ersten Teil des Themennamens bildet. Die Themenzeichenfolge in *topicObject* ist mit *topicName* verkettet. Die Regeln für die Erstellung von Themenzeichenfolgen sind im Abschnitt Themenzeichenfolgen kombinieren definiert.

Bei der Ausgabe enthält *topicObject* den Namen des Verwaltungsthemenobjekts, das die größte Übereinstimmung in der Themenstruktur mit dem durch den Themennamen angegebenen Thema aufweist.

openAs
Greift auf das Thema für eine Veröffentlichung oder Subskription zu. Der Parameter kann nur eine dieser Optionen enthalten:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

Kombinieren Sie die Optionen, mit denen das Öffnen des Themas entweder für die Veröffentlichung oder für die Subskription gesteuert wird. Mit den MQC.MQSO_*-Konstanten können Sie auf ein Thema für die Subskription zugreifen, mit den MQC.MQOO_*-Konstanten können Sie auf ein Thema für die Veröffentlichung zugreifen.

Wenn mehrere Optionen erforderlich sind, addieren Sie die Werte zusammen oder kombinieren Sie die Optionswerte mit dem bitweisen Operator OR .

alternateUserId
Geben Sie die alternative Benutzer-ID an, die zur Überprüfung der erforderlichen Berechtigung zum Beenden des Vorgangs erforderlich ist. Sie müssen *alternateUserId* angeben, wenn entweder MQC.MQOO_ALTERNATE_USER_AUTHORITY oder MQC.MQSO_ALTERNATE_USER_AUTHORITY im Optionsparameter festgelegt ist.

subscriptionName
subscriptionName ist erforderlich, wenn die Option MQC.MQSO_DURABLE oder MQC.MQSO_ALTER bereitgestellt wird. In beiden Fällen ist MQTopic implizit für die Subskription geöffnet. Eine Ausnahme wird ausgelöst, wenn MQC.MQSO_DURABLE festgelegt ist und die Subskription vorhanden ist oder wenn MQC.MQSO_ALTER festgelegt ist und die Subskription nicht vorhanden ist.

properties
Legen Sie eine der besonderen Subskriptionseigenschaften fest, die mithilfe einer Hashtabelle aufgelistet wurden. In der Hashtabelle angegebene Einträge werden mit Ausgabewerten aktualisiert. Einträge werden nicht zur Hashtabelle hinzugefügt, um Berichte für Ausgabewerte zu erstellen.

- 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

.NET-Schnittstelle IMQObjectTrigger

Implementieren Sie IMQObjectTrigger, um Nachrichten zu verarbeiten, die vom .NET-Monitor **runmqdnm** übergeben wurden.

Schnittstelle

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

Abhängig davon, ob eine Synchronisationspunktsteuerung im Befehl **runmqdnm** angegeben wurde, wird die Nachricht vor oder nach der Rückgabe der Execute-Methode aus der Warteschlange entfernt.

Methoden

void Execute (MQQueueManager *queueManager*, MQQueue *queue*, MQMessage *message*, string *param*);

queueManager
Der Warteschlangenmanager, der als Host für die überwachte Warteschlange dient.

queue
Die überwachte Warteschlange.

message
Die aus der Warteschlange gelesene Nachricht.

param
Von UserParameter übergebene Daten.

.NET-Schnittstelle MQC

Verweisen Sie auf eine MQI-Konstante durch Voranstellen von MQC. vor den Konstantennamen. MQC definiert alle Konstanten, die von MQI verwendet werden.

Schnittstelle

```
System.Object
    │
    └─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Beispiel

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

Zeichensatzkennungen für .NET-Anwendungen

Beschreibungen der Zeichensätze, die Sie für die Codierung von .NET IBM WebSphere MQ-Nachrichten auswählen können

Zeichensatz	Beschreibung
273	ibm037

Zeichensatz	Beschreibung
437	ibm437 / PC-Vorlage
500	ibm500
819	iso-8859-1 / latin1 / ibm819
1200	Unicode
1208	UTF-8
273	ibm273
277	ibm277
278	ibm278
280	ibm280
284	ibm284
285	ibm285
297	ibm297
420	ibm420
424	ibm424
737	ibm737 / PC-Griechisch
775	ibm775 / PC-Baltisch
813	iso-8859-7 / Griechisch / ibm813
838	ibm838
850	ibm850 / PC-Latein 1
852	ibm852 / PC-Latein 2
855	ibm855 / PC-Kyrillisch
856	ibm856
857	ibm857 / PC-Türkisch
860	ibm860 / PC-Portugiesisch
861	ibm861 / PC-Isländisch
862	ibm862 / PC-Hebräisch
863	ibm863 / PC-Französisch (Kanada)
864	ibm864 / PC-Arabisch
865	ibm865 / PC-Nordländer
866	ibm866 / PC-Russisch
868	ibm868
869	ibm869 / PC-Modern-Griechisch
870	ibm870
871	ibm871
874	ibm874

Zeichensatz	Beschreibung
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 / Kyrillisch / ibm915
916	iso-8859-8 / Hebräisch / ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930
932	PC-Japanisch
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 / Big 5 traditionelles Chinesisch
954	EUCJIS
964	ibm964 / CNS 11643 traditionelles Chinesisch
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 / Arabisch / ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Lateinisch 2

Zeichensatz	Beschreibung
1251	Windows Kyrillisch
1252	Windows Lateinisch 1
1253	Windows Griechisch
1254	Windows Türkisch
1255	Windows Hebräisch
1256	Windows Arabisch
1257	Windows Baltisch
1258	Windows Vietnamesisch
1381	ibm1381
1383	ibm1383
2022	JIS
5601	ksc-5601 Koreanisch
33722	ibm33722

IBM WebSphere MQ-C++-Klassen

Die IBM WebSphere MQ-C++-Klassen binden die IBM WebSphere MQ Message Queue Interface (MQI) mit ein. Die einzelne C++-Headerdatei **imqi.hpp** deckt all diese Klassen ab.

Für jede Klasse werden folgende Informationen angezeigt:

Klassenhierarchiediagramm
Ein Klassendiagramm, das die Vererbungsbeziehung der Klasse zu seinen direkt übergeordneten Klassen anzeigt, falls vorhanden.

Andere relevante Klassen
Dokumentlinks zu anderen relevanten Klassen, z. B. übergeordneten Klassen und den Klassen von Objekten, die in Methodensignaturen verwendet werden.

Objektattribute
Attribute der Klasse. Sie werden zusätzlich zu den Attributen für übergeordnete Klassen definiert. Viele Attribute geben WebSphere MQ-Datenstrukturelemente an (siehe „Querverweise zwischen C++ und MQI" auf Seite 1345). Ausführliche Beschreibungen finden Sie unter „Attribute von Objekten" auf Seite 797.

Konstruktoren
Signaturen der speziellen Methoden, mit denen ein Objekt der Klasse erstellt wird.

Objektmethoden (öffentlich)
Signaturen von Methoden, die für die Operation eine Instanz der Klasse erfordern und die keinen Nutzungsbeschränkungen unterliegen.

Gegebenenfalls werden auch folgende Informationen angezeigt:

Klassenmethoden (öffentlich)
Signaturen von Methoden, die für die Operation keine Instanz der Klasse erfordern und die keinen Nutzungsbeschränkungen unterliegen.

Überlastete Methoden (übergeordneter Klassen)
Signaturen dieser virtuellen Methoden, die in übergeordneten Klassen definiert sind, jedoch ein anderes, polymorphes Verhalten für diese Klasse aufweisen.

Objektmethoden (geschützt)
Signaturen von Methoden, die für die Operation eine Instanz der Klasse erfordern und die für die Verwendung durch die Implementierung abgeleiteter Klassen reserviert sind. Dieser Abschnitt ist nur für Klassenautoren von Interesse, nicht für Klassenbenutzer.

Objektdaten (geschützt)
Implementationsdetails für Objektdaten, die für die Implementierungen abgeleiteter Klassen verfügbar sind. Dieser Abschnitt ist nur für Klassenautoren von Interesse, nicht für Klassenbenutzer.

Ursachencodes
MQRC_*-Werte (siehe API-Ursachencodes), die von diesen Methoden, die fehlschlagen, erwartet werden können. Eine vollständige Liste der Ursachencodes, die für ein Objekt einer Klasse auftreten können, finden Sie in der Dokumentation für übergeordnete Klassen. Die dokumentierte Liste der Ursachencodes für eine Klasse enthält nicht die Ursachencodes für übergeordnete Klassen.

Anmerkung:

1. Objekte dieser Klassen sind nicht threadsicher. Dadurch wird eine optimale Leistung gewährleistet. Achten Sie jedoch darauf, dass Sie immer nur von einem Thread aus auf ein Objekt zugreifen.
2. Es wird empfohlen, bei einem Multithreadprogramm für jeden Thread ein separates ImqQueueManager-Objekt zu verwenden. Jedes Manager-Objekt muss eine eigene unabhängige Sammlung anderer Objekte haben, um sicherzustellen, dass Objekte in verschiedenen Threads voneinander getrennt werden.

Folgende Klassen sind verfügbar:

- „C++-Klasse "ImqAuthenticationRecord"” auf Seite 1359
- „C++-Klasse "ImqBinary"” auf Seite 1361
- „C++-Klasse "ImqCache"” auf Seite 1363
- „C++-Klasse "ImqChannel"” auf Seite 1367
- „C++-Klasse "ImqCICSBridgeHeader"” auf Seite 1373
- „C++-Klasse "ImqDeadLetterHeader"” auf Seite 1379
- „C++-Klasse "ImqDistributionList"” auf Seite 1382
- „C++-Klasse "ImqError"” auf Seite 1383
- „C++-Klasse "ImqGetMessageOptions"” auf Seite 1384
- „C++-Klasse "ImqHeader"” auf Seite 1388
- „C++-Klasse "ImqIMSBridgeHeader"” auf Seite 1389
- „C++-Klasse "ImqItem"” auf Seite 1392
- „C++-Klasse "ImqMessage"” auf Seite 1393
- „C++-Klasse "ImqMessageTracker"” auf Seite 1400
- „C++-Klasse "ImqNamelist"” auf Seite 1404
- „C++-Klasse "ImqObject"” auf Seite 1405
- „C++-Klasse "ImqProcess"” auf Seite 1411
- „C++-Klasse "ImqPutMessageOptions"” auf Seite 1412
- „C++-Klasse "ImqQueue"” auf Seite 1415
- „C++-Klasse "ImqQueueManager"” auf Seite 1427
- „C++-Klasse "ImqReferenceHeader"” auf Seite 1445
- „C++-Klasse ImqString” auf Seite 1448
- „C++-Klasse ImqTrigger” auf Seite 1453
- „C++-Klasse ImqWorkHeader” auf Seite 1456

Querverweise zwischen C++ und MQI

In dieser Themengruppe finden Sie Informationen zu Querverweisen zwischen C++ und MQI.

Hinweise finden Sie in diesem Abschnitt und im Abschnitt „Im MQI verwendete Datentypen” auf Seite 217.

In dieser Tabelle finden Sie die Beziehungen zwischen MQI-Datenstrukturen und den C++-Klassen und Include-Dateien. In den folgenden Abschnitten werden Informationen zu Querverweisen für jede C++-Klasse dargestellt. Diese Querverweise beziehen sich auf die Verwendung der zugrunde liegenden prozeduralen Schnittstellen von WebSphere MQ. Die Klassen ImqBinary, ImqDistributionList und ImqString weisen keine Attribute auf, die in diese Kategorie fallen, und werden ausgeschlossen.

Tabelle 610. Querverweise zwischen Datenstruktur, Klasse und Include-Datei

Datenstruktur	**Klasse**	**Include-Datei**
MQAIR	ImqAuthenticationRecord	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH	ImqCICSBridgeHeader	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	ImqDistributionList	imqdst.hpp
	ImqError	imqerr.hpp
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	ImqIMSBridgeHeader	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	ImqMessageTracker	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	ImqQueueManager	imqmgr.hpp
MQRMH	ImqReferenceHeader	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	ImqWorkHeader	imqwih.hpp

Querverweise für ImqAuthenticationRecord

Querverweise zwischen Attributen, Datenstrukturen, Feldern und Aufrufen für die C++-Klasse ImqAuthenticationRecord.

Attribut	Datenstruktur	Feld	Aufruf
Verbindungsname (connection name)	MQAIR	AuthInfoConnName	MQCONNX
Kennwort	MQAIR	LDAPPassword	MQCONNX
Typ	MQAIR	AuthInfoType	MQCONNX
Benutzername	MQAIR	LDAPUserNamePtr	MQCONNX
	MQAIR	LDAPUserNameOffset	MQCONNX
	MQAIR	LDAPUserNameLength	MQCONNX

Querverweise für ImqCache

Querverweise zwischen Attributen und Aufrufen für die C++-Klasse ImqCache.

Attribut	Aufruf
automatic buffer	MQGET
buffer length	MQGET
buffer pointer	MQGET, MQPUT
data length	MQGET
data offset	MQGET
data pointer	MQGET
Nachrichtenlänge	MQGET, MQPUT

Querverweise für ImqChannel

Querverweise zwischen Attributen, Datenstrukturen, Feldern und Aufrufen für die C++-Klasse ImqChannel.

Attribut	Datenstruktur	Feld	Aufruf
batch heart-beat	MQCD	BatchHeartbeat	MQCONNX
Kanalname	MQCD	ChannelName	MQCONNX
Verbindungsname (connection name)	MQCD	ConnectionName	MQCONNX
	MQCD	ShortConnectionName	MQCONNX
Headerkomprimierung	MQCD	HdrCompList	MQCONNX
heart-beat interval	MQCD	HeartbeatInterval	MQCONNX
keep alive interval	MQCD	KeepAliveInterval	MQCONNX
lokale Adresse	MQCD	LocalAddress	MQCONNX
maximale Nachrichtenlänge	MQCD	MaxMsgLength	MQCONNX
message compression	MQCD	MsgCompList	MQCONNX

Attribut	Datenstruk-tur	Feld	Aufruf
Modusname	MQCD	ModeName	MQCONNX
Kennwort	MQCD	Passwort	MQCONNX
receive exit count	MQCD		MQCONNX
receive exit names	MQCD	ReceiveExit	MQCONNX
	MQCD	ReceiveExitsDefined	MQCONNX
	MQCD	ReceiveExitPtr	MQCONNX
receive user data	MQCD	ReceiveUserData	MQCONNX
	MQCD	ReceiveUserDataPtr	MQCONNX
Name des Sicherheitsexits	MQCD	SecurityExit	MQCONNX
security user data	MQCD	SecurityUserData	MQCONNX
send exit count	MQCD		MQCONNX
send exit names	MQCD	SendExit	MQCONNX
	MQCD	SendExitsDefined	MQCONNX
	MQCD	SendExitPtr	MQCONNX
send user data	MQCD	SendUserData	MQCONNX
	MQCD	SendUserDataPtr	MQCONNX
SSL-Verschl.spezifikation	MQCD	sslCipherSpecification	MQCONNX
SSL client authentication type	MQCD	sslClientAuthentication	MQCONNX
SSL-Peer-Name	MQCD	sslPeerName	MQCONNX
transaction program name	MQCD	TpName	MQCONNX
Transporttyp	MQCD	TransportType	MQCONNX
Benutzer-ID	MQCD	UserIdentifier	MQCONNX

Querverweise für ImqCICSBridgeHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqCICSBridgeHeader.

Attribut	Datenstruktur	Feld
bridge abend code	MQCIH	AbendCode
ADS descriptor	MQCIH	AdsDescriptor
attention identifier	MQCIH	AttentionId
authenticator	MQCIH	Authenticator
bridge completion code	MQCIH	BridgeCompletionCode
bridge error offset	MQCIH	ErrorOffset
bridge reason code	MQCIH	BridgeReason
bridge cancel code	MQCIH	CancelCode

Attribut	Datenstruktur	Feld
conversational task	MQCIH	ConversationalTask
cursor position	MQCIH	CursorPosition
facility token	MQCIH	Funktion
facility keep time	MQCIH	FacilityKeepTime
facility like	MQCIH	FacilityLike
function	MQCIH	Funktion
get wait interval	MQCIH	GetWaitInterval
link type	MQCIH	LinkType
next transaction identifier	MQCIH	NextTransactionId
output data length	MQCIH	OutputDataLength
reply-to format	MQCIH	ReplyToFormat
bridge return code	MQCIH	ReturnCode
start code	MQCIH	StartCode
task end status	MQCIH	TaskEndStatus
transaction identifier	MQCIH	TransactionId
uow control	MQCIH	UowControl
Version	MQCIH	Version

Querverweise für ImqDeadLetterHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqDeadLetterHeader.

Attribut	Datenstruktur	Feld
dead-letter reason code	MQDLH	Ursache
destination queue manager name	MQDLH	DestQMgrName
destination queue name	MQDLH	DestQName
put application name	MQDLH	PutApplName
put application type	MQDLH	PutApplType
put date	MQDLH	PutDate
put time	MQDLH	PutTime

Querverweise für ImqError

Querverweise zwischen Attributen und Aufrufen für die C++-Klasse ImqError.

Attribut	Aufruf
Beendigungscode	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET
Ursachencode	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

Querverweise für ImqGetMessageOptions

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqGetMessageOptions.

Attribut	Datenstruktur	Feld
group status	MQGMO	GroupStatus
match options	MQGMO	MatchOptions
Nachrichten-Token	MQGMO	MessageToken
Optionen	MQGMO	Optionen
resolved queue name	MQGMO	ResolvedQName
returned length	MQGMO	ReturnedLength
Segmentierung	MQGMO	Segmentation
segment status	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
syncpoint participation	MQGMO	Optionen
wait interval	MQGMO	WaitInterval

Querverweise für ImqHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqHeader.

Attribut	Datenstruktur	Feld
character set	MQDLH, MQIIH	CodedCharSetId
encoding	MQDLH, MQIIH	Encoding
Format	MQDLH, MQIIH	Format
header flags	MQIIH, MQRMH	Markierungen

Querverweise für ImqIMSBridgeHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqReferenceHeader.

Attribut	Datenstruktur	Feld
authenticator	MQIIH	Authenticator
commit mode	MQIIH	CommitMode
logical terminal override	MQIIH	LTermOverride
message format services map name	MQIIH	MFSMapName
reply-to format	MQIIH	ReplyToFormat
security scope	MQIIH	SecurityScope
transaction instance id	MQIIH	TranInstanceId
transaction state	MQIIH	TranState

Querverweise für ImqItem

Querverweise zwischen Attributen und Aufrufen für die C++-Klasse ImqItem.

Attribut	Aufruf
structure id	MQGET

Querverweise für ImqMessage

Querverweise zwischen Attributen, Datenstrukturen, Feldern und Aufrufen für die C++-Klasse ImqMessage.

Attribut	Datenstruktur	Feld	Aufruf
application id data	MQMD	ApplIdentityData	
application origin data	MQMD	ApplOriginData	
backout count	MQMD	BackoutCount	
character set	MQMD	CodedCharSetId	
encoding	MQMD	Encoding	
expiry	MQMD	Verfall	
Format	MQMD	Format	
message flags	MQMD	MsgFlags	
Nachrichtentyp	MQMD	MsgType	
offset	MQMD	Offset	
original length	MQMD	OriginalLength	
persistence	MQMD	Permanenz	
priority	MQMD	Priority	
put application name	MQMD	PutApplName	
put application type	MQMD	PutApplType	
put date	MQMD	PutDate	
put time	MQMD	PutTime	
reply-to queue manager name	MQMD	ReplyToQMgr	
reply-to queue name	MQMD	ReplyToQ	
Bericht	MQMD	Bericht	
sequence number	MQMD	MsgSeqNumber	
total message length		DataLength	MQGET
Benutzer-ID	MQMD	UserIdentifier	

Querverweise für ImqMessageTracker

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqMessageTracker.

Attribut	Datenstruktur	Feld
accounting token	MQMD	AccountingToken

Attribut	Datenstruktur	Feld
correlation id	MQMD	CorrelId
feedback	MQMD	Feedback
group id	MQMD	GroupId
message id	MQMD	MsgId

Querverweise für ImqNamelist

Querverweise zwischen Attributen, Abfragen und Aufrufen für die C++-Klasse ImqNamelist.

Attribut	Anfrage	Aufruf
name count	MQIA_NAME_COUNT	MQINQ
namelist name	MQCA_NAMELIST_NAME	MQINQ

Querverweise für ImqObject

Querverweise zwischen Attributen, Datenstrukturen, Feldern, Abfragen und Aufrufen für die C++-Klasse ImqObject.

Attribut	Daten-struktur	Feld	Anfrage	Aufruf
alteration date			MQCA_ALTERATION_DATE	MQINQ
alteration time			MQCA_ALTERATION_TIME	MQINQ
Alternative Benutzer-ID	MQOD	AlternateUserId		
alternate security id				
close options				MQCLOSE
Beschreibung			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
Name	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PRO-CESS_NAME	MQINQ
open options				MQOPEN
open status				MQOPEN, MQCLOSE
queue manager iden-tifier	queue manager identifier		MQCA_Q_MGR_IDENTIFIER	MQINQ

Querverweise für ImqProcess

Querverweise zwischen Attributen, Abfragen und Aufrufen für die C++-Klasse ImqProcess.

Attribut	Anfrage	Aufruf
application id	MQCA_APPL_ID	MQINQ
application type	MQIA_APPL_TYPE	MQINQ
environment data	MQCA_ENV_DATA	MQINQ

Attribut	Anfrage	Aufruf
Benutzerdaten	MQCA_USER_DATA	MQINQ

Querverweise für ImqPutMessageOptions

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqReferenceHeader.

Tabelle 611. Querverweise für ImqPutMessageOptions

Attribut	Datenstruktur	Feld
context reference	MQPMO	Context
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
Optionen	MQPMO	Optionen
record fields	MQPMO	PutMsgRecFields
resolved queue manager name	MQPMO	ResolvedQMgrName
resolved queue name	MQPMO	ResolvedQName
	MQPMO	Zeitlimit
	MQPMO	UnknownDestCount
syncpoint participation	MQPMO	Optionen

Querverweise für ImqQueue

Querverweise zwischen Attributen, Datenstrukturen, Feldern, Abfragen und Aufrufen für die C++-Klasse ImqQueue.

Tabelle 612. Querverweise für ImqQueue

Attribut	Datenstruktur	Feld	Anfrage	Aufruf
backout requeue name			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
backout threshold			MQIA_BACKOUT_THRESHOLD	MQINQ
base queue name			MQCA_BASE_Q_NAME	MQINQ
Clustername			MQCA_CLUSTER_NAME	MQINQ
cluster namelist name			MQCA_CLUSTER_NAMELIST	MQINQ
cluster workload rank			MQIA_CLWL_Q_RANK	MQINQ
cluster workload priority			MQIA_CLWL_Q_PRIORITY	MQINQ
cluster workload use queue			MQIA_CLWL_USEQ	MQINQ
creation date			MQCA_CREATION_DATE	MQINQ
creation time			MQCA_CREATION_TIME	MQINQ

Tabelle 612. Querverweise für ImqQueue (Forts.)

Attribut	Datenstruktur	Feld	Anfrage	Aufruf
current depth			MQIA_CURRENT_Q_DEPTH	MQINQ
default bind			MQIA_DEF_BIND	MQINQ
default input open option			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
default persistence			MQIA_DEF_PERSISTENCE	MQINQ
default priority			MQIA_DEF_PRIORITY	MQINQ
definition type			MQIA_DEFINITION_TYPE	MQINQ
depth high event			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
depth high limit			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
depth low event			MQIA_Q_DEPTH_LOW_EVENT	MQINQ
depth low limit			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
depth maximum event			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
distribution lists			MQIA_DIST_LISTS	MQINQ, MQSET
dynamic queue name	MQOD	DynamicQName		
harden get backout			MQIA_HARDEN_GET_BACKOUT	MQINQ
index type			MQIA_INDEX_TYPE	MQINQ
inhibit get			MQIA_INHIBIT_GET	MQINQ, MQSET
inhibit put			MQIA_INHIBIT_PUT	MQINQ, MQSET
initiation queue name			MQCA_INITIATION_Q_NAME	MQINQ
maximum depth			MQIA_MAX_Q_DEPTH	MQINQ
maximale Nachrichtenlänge			MQIA_MAX_MSG_LENGTH	MQINQ
Reihenfolge bei der Nachrichtenübertragung			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
next distributed queue				
non persistent message class			MQIA_NPM_CLASS	MQINQ
open input count			MQIA_OPEN_INPUT_COUNT	MQINQ
open output count			MQIA_OPEN_OUTPUT_COUNT	MQINQ
previous distributed queue				
process name			MQCA_PROCESS_NAME	MQINQ

Tabelle 612. Querverweise für ImqQueue (Forts.)

Attribut	Datenstruktur	Feld	Anfrage	Aufruf
queue accounting			MQIA_ACCOUNTING_Q	MQINQ
Name des Warteschlangenmanagers	MQOD	ObjectQMgrName		
queue monitoring			MQIA_MONITORING_Q	MQINQ
queue statistics			MQIA_STATISTICS_Q	MQINQ
Warteschlangentyp			MQIA_Q_TYPE	MQINQ
Ferner Warteschlangenmanagername			MQCA_REMOTE_Q_MGR_NAME	MQINQ
remote queue name			MQCA_REMOTE_Q_NAME	MQINQ
resolved queue manager name	MQOD	ResolvedQMgrName		
resolved queue name	MQOD	ResolvedQName		
retention interval			MQIA_RETENTION_INTERVAL	MQINQ
Geltungsbereich			MQIA_SCOPE	MQINQ
Serviceintervall			MQIA_Q_SERVICE_INTERVAL	MQINQ
Serviceintervallereignis			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
shareability			MQIA_SHAREABILITY	MQINQ
Speicherklasse			MQCA_STORAGE_CLASS	MQINQ
Name der Übertragungswarteschlange			MQCA_XMIT_Q_NAME	MQINQ
trigger control			MQIA_TRIGGER_CONTROL	MQINQ, MQSET
trigger data			MQCA_TRIGGER_DATA	MQINQ, MQSET
trigger depth			MQIA_TRIGGER_DEPTH	MQINQ, MQSET
trigger message priority			MQIA_TRIGGER_MSG_PRIORITY	MQINQ, MQSET
trigger type			MQIA_TRIGGER_TYPE	MQINQ, MQSET
Belegung			MQIA_USAGE	MQINQ

Querverweise für ImqQueueManager

Querverweise zwischen Attributen, Datenstrukturen, Feldern, Abfragen und Aufrufen für die C++-Klasse ImqQueueManager.

Attribut	Datenstruktur	Feld	Anfrage	Aufruf
accounting connections override			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
accounting interval			MQIA_ACCOUNTING_INTERVAL	MQINQ
activity recording			MQIA_ACTIVITY_RECORDING	MQINQ
adopt new mca check			MQIA_ADOPTNEWMCA_CHECK	MQINQ
adopt new mca type			MQIA_ADOPTNEWMCA_TYPE	MQINQ
authentication type	MQCSP	AuthenticationType		MQCONNX
authority event			MQIA_AUTHORITY_EVENT	MQINQ
begin options	MQBO	Optionen		MQBEGIN
bridge event			MQIA_BRIDGE_EVENT	MQINQ
channel auto definition			MQIA_CHANNEL_AUTO_DEF	MQINQ
channel auto definition event			MQIA_CHANNEL_AUTO_EVENT	MQIA
channel auto definition exit			MQIA_CHANNEL_AUTO_EXIT	MQIA
channel event			MQIA_CHANNEL_EVENT	MQINQ
channel initiator adapters			MQIA_CHINIT_ADAPTERS	MQINQ
channel initiator control			MQIA_CHINIT_CONTROL	MQINQ
channel initiator dispatchers			MQIA_CHINIT_DISPATCHERS	MQINQ
channel initiator trace auto start			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
channel initiator trace table size			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
channel monitoring			MQIA_MONITORING_CHANNEL	MQINQ
channel reference	MQCD	ChannelType		MQCONNX
Kanalstatistik			MQIA_STATISTICS_CHANNEL	MQINQ
character set			MQIA_CODED_CHAR_SET_ID	MQINQ
cluster sender monitoring			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
cluster sender statistics			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
cluster workload data			MQCA_CLUSTER_WORKLOAD_DATA	MQINQ

Attribut	Daten-struktur	Feld	Anfrage	Aufruf
cluster workload exit			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
cluster workload length			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
cluster workload mru			MQIA_CLWL_MRU_CHANNELS	MQINQ
cluster workload use queue			MQIA_CLWL_USEQ	MQINQ
command event			MQIA_COMMAND_EVENT	MQINQ
command input queue name			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
command level			MQIA_COMMAND_LEVEL	MQINQ
command server control			MQIA_CMD_SERVER_CONTROL	MQINQ
connect options	MQCNO	Optionen		MQCONN, MQCONNX
connection id	MQCNO	ConnectionId		MQCONNX
connection status				MQCONN, MQCONNX, MQDISC
connection tag	MQCD	ConnTag		MQCONNX
cryptographic hardware	MQSCO	CryptoHardware		MQCONNX
dead-letter queue name			MQCA_DEAD_LETTER_Q_NAME	MQINQ
default transmissi-on queue name			MQCA_DEF_XMIT_Q_NAME	MQINQ
distribution lists			MQIA_DIST_LISTS	MQINQ
dns group			MQCA_DNS_GROUP	MQINQ
dns wlm			MQIA_DNS_WLM	MQINQ
first authentication record	MQSCO	AuthInfoRecOffset		MQCONNX
	MQSCO	AuthInfoRecPtr		MQCONNX
inhibit event			MQIA_INHIBIT_EVENT	MQINQ
ip address version			MQIA_IP_ADDRESS_VERSION	MQINQ
Schlüsselreposito-ry	MQSCO	KeyRepository		MQCONNX
key reset count	MQSCO	KeyResetCount		MQCONNX
listener timer			MQIA_LISTENER_TIMER	MQINQ
local event			MQIA_LOCAL_EVENT	MQINQ

Attribut	Daten-struktur	Feld	Anfrage	Aufruf
logger event			MQIA_LOGGER_EVENT	MQINQ
lu group name			MQCA_LU_GROUP_NAME	MQINQ
lu name			MQCA_LU_NAME	MQINQ
lu62 arm suffix			MQCA_LU62_ARM_SUFFIX	MQINQ
lu62 channels			MQIA_LU62_CHANNELS	MQINQ
maximum active channels			MQIA_ACTIVE_CHANNELS	MQINQ
maximum chan-nels			MQIA_MAX_CHANNELS	MQINQ
maximum handles			MQIA_MAX_HANDLES	MQINQ
maximale Nach-richtenlänge			MQIA_MAX_MSG_LENGTH	MQINQ
maximum priority			MQIA_MAX_PRIORITY	MQINQ
maximum uncom-mitted messages			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
mqi accounting			MQIA_ACCOUNTING_MQI	MQINQ
mqi statistics			MQIA_STATISTICS_MQI	MQINQ
outbound port ma-ximum			MQIA_OUTBOUND_PORT_MAX	MQINQ
outbound port mi-nimum			MQIA_OUTBOUND_PORT_MIN	MQINQ
Kennwort	MQCSP	CSPPasswordPtr		MQCONNX
	MQCSP	CSPPasswordOffset		MQCONNX
	MQCSP	CSPPassword-Length		MQCONNX
Leistungsereignis			MQIA_PERFORMANCE_EVENT	MQINQ
platform			MQIA_PLATFORM	MQINQ
queue accounting			MQIA_ACCOUNTING_Q	MQINQ
queue monitoring			MQIA_MONITORING_Q	MQINQ
queue statistics			MQIA_STATISTICS_Q	MQINQ
receive timeout			MQIA_RECEIVE_TIMEOUT	MQINQ
receive timeout mi-nimum			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
receive timeout ty-pe			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
remote event			MQIA_REMOTE_EVENT	MQINQ
repository name			MQCA_REPOSITORY_NAME	MQINQ
repository namelist			MQCA_REPOSITORY_NAMELIST	MQINQ

Attribut	Daten-struktur	Feld	Anfrage	Aufruf
shared queue queue manager name			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
ssl event			MQIA_SSL_EVENT	MQINQ
ssl fips			MQIA_SSL_FIPS_REQUIRED	MQINQ
ssl key reset count			MQIA_SSL_RESET_COUNT	MQINQ
start-stop event			MQIA_START_STOP_EVENT	MQINQ
statistics interval			MQIA_STATISTICS_INTERVAL	MQINQ
syncpoint availabi-lity			MQIA_SYNCPOINT	MQINQ
tcp channels			MQIA_TCP_CHANNELS	MQINQ
tcp keep alive			MQIA_TCP_KEEP_ALIVE	MQINQ
tcp name			MQCA_TCP_NAME	MQINQ
tcp stack type			MQIA_TCP_STACK_TYPE	MQINQ
trace route recor-ding			MQIA_TRACE_ROUTE_RECORDING	MQINQ
trigger interval			MQIA_TRIGGER_INTERVAL	MQINQ
Benutzer-ID	MQCSP	CSPUserIdPtr		MQCONNX
	MQCSP	CSPUserIdOffset		MQCONNX
	MQCSP	CSPUserIdLength		MQCONNX

Querverweise für ImqReferenceHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqReferenceHe-ader.

Attribut	Datenstruktur	Feld
destination environment	MQRMH	DestEnvLength, DestEnvOffset
destination name	MQRMH	DestNameLength, DestNameOffset
instance id	MQRMH	ObjectInstanceId
logical length	MQRMH	DataLogicalLength
logical offset	MQRMH	DataLogicalOffset
logical offset 2	MQRMH	DataLogicalOffset2
reference type	MQRMH	ObjectType
source environment	MQRMH	SrcEnvLength, SrcEnvOffset
source name	MQRMH	SrcNameLength, SrcNameOffset

Querverweise für ImqTrigger

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqReferenceHeader.

Tabelle 613. Querverweise für ImqTrigger

Attribut	Datenstruktur	Feld
application id	MQTM	ApplId
application type	MQTM	ApplType
environment data	MQTM	EnvData
process name	MQTM	ProcessName
Warteschlangenname	MQTM	QName
trigger data	MQTM	TriggerData
Benutzerdaten	MQTM	UserData

Querverweise für ImqWorkHeader

Querverweise zwischen Attributen, Datenstrukturen und Feldern für die C++-Klasse ImqReferenceHeader.

Attribut	Datenstruktur	Feld
Nachrichten-Token	MQWIH	MessageToken
Servicename	MQWIH	ServiceName
service step	MQWIH	ServiceStep

C++-Klasse "ImqAuthenticationRecord"

Diese Klasse bindet einen Datensatz mit Authentifizierungsdaten (MQAIR) ein, der bei Ausführung der Methode "ImqQueueManager::connect" für benutzerdefinierte SSL-Clientverbindungen verwendet werden soll.

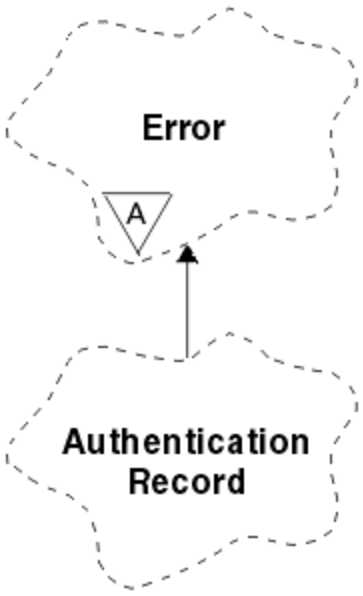


Abbildung 46. ImqAuthenticationRecord-Klasse

Weitere Informationen finden Sie in der Beschreibung der Methode "ImqQueueManager::connect". Diese Klasse ist auf der z/OS-Plattform nicht verfügbar.

Objektattribute

Verbindungsname (connection name)
Der Name der Verbindung mit dem LDAP-CRL-Server. Hierbei handelt es sich um die IP-Adresse oder den DNS-Namen ggf. gefolgt von der Portnummer in Klammern.

Verbindungsverweis (connection reference)
Ein Verweis auf ein ImqQueueManager-Objekt, das die erforderliche Verbindung zu einem (lokalen) Warteschlangenmanager bereitstellt. Der Anfangswert ist null. Verwechseln Sie dies nicht mit dem Warteschlangenmanagernamen, der einen (möglicherweise fernen) Warteschlangenmanager für eine benannte Warteschlange angibt.

nächster Authentifizierungsdatensatz (next authentication record)
Nächstes Objekt dieser Klasse in keiner bestimmten Reihenfolge mit demselben **Verbindungsverweis** wie dieses Objekt. Der Anfangswert ist null.

Kennwort
Ein dem LDAP-CRL-Server für die Verbindungsauthentifizierung bereitgestelltes Kennwort.

vorheriger Authentifizierungsdatensatz (previous authentication record)
Vorheriges Objekt dieser Klasse in keiner bestimmten Reihenfolge mit demselben **Verbindungsverweis** wie dieses Objekt. Der Anfangswert ist null.

Typ
Der im Datensatz enthaltene Authentifizierungsdatentyp.

Benutzername
Eine dem LDAP-CRL-Server zur Berechtigungsprüfung bereitgestellte Benutzer-ID.

Konstruktoren

ImqAuthenticationRecord( );
Der Standardkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqAuthentication-Datensatz & *Luft*);
Kopiert Instanzdaten aus *air* und ersetzt dabei die bereits vorhandenen Instanzdaten.

const ImqString & connectionName ( ) const ;
Gibt den Verbindungsnamen (**connection name**) zurück.

void setConnectionName (const ImqString & *Name*);
Legt den Verbindungsnamen (**connection name**) fest.

void setConnectionName (const char * *name* = 0);
Legt den Verbindungsnamen (**connection name**) fest.

ImqQueueManager * connectionReference ( ) const ;
Gibt den Verbindungsverweis (**connection reference**) zurück.

void setConnectionReference (ImqQueueManager & *manager*)
Legt den Verbindungsverweis (**connection reference**) fest.

void setConnectionReference (ImqQueueManager * *manager* = 0);
Legt den Verbindungsverweis (**connection reference**) fest.

void copyOut (MQAIR * *pAir*);
Kopiert Instanzdaten in *pAir* und ersetzt dabei die bereits vorhandenen Instanzdaten. Dabei muss möglicherweise abhängiger Speicher zugeordnet werden.

void clear (MQAIR * *pAir*);
Löscht die Struktur und gibt abhängigen Speicher frei, auf den *pAir* verweist.

ImqAuthenticationRecord * nextAuthenticationRecord ( ) const ;
Gibt den nächsten Authentifizierungsdatensatz (**next authentication record**) zurück.

const ImqString & password ( ) const ;
Gibt das Kennwort (**password**) zurück.

void setPassword (const ImqString & *Kennwort*);
Legt das Kennwort (**password**) fest.

void setPassword (const char * *password* = 0);
Legt das Kennwort (**password**) fest.

ImqAuthenticationRecord * previousAuthenticationRecord ( ) const ;
Gibt den vorherigen Authentifizierungsdatensatz (**previous authentication record**) zurück.

MQLONG type ( ) const ;
Gibt den Typ (**type**) zurück.

void setType (const MQLONG *type*);
Setzt den Typ (**type**).

const ImqString & userName ( ) const ;
Gibt den Benutzernamen (**user name**) zurück.

void setUserName (const ImqString & *Name*);
Setzt den Benutzernamen (**user name**).

void setUserName (const char * *name* = 0);
Setzt den Benutzernamen (**user name**).

Objektmethoden (geschützt)

void setNextAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Setzt den nächsten Authentifizierungsdatensatz (**next authentication record**).

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass damit die Liste mit den Authentifizierungdatensätzen nicht beschädigt wird.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Setzt den vorherigen Authentifizierungsdatensatz (**previous authentication record**).

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass damit die Liste mit den Authentifizierungdatensätzen nicht beschädigt wird.

C++-Klasse "ImqBinary"

Diese Klasse bindet eine binäre Bytefeldgruppe ein, die für "ImqMessage"-Werte für Abrechnungstoken (**accounting token**), Korrelations-ID (**correlation id**) und Nachrichten-ID (**message id**) verwendet werden kann. Sie ermöglicht ein einfaches Zuordnen, Kopieren und Vergleichen.

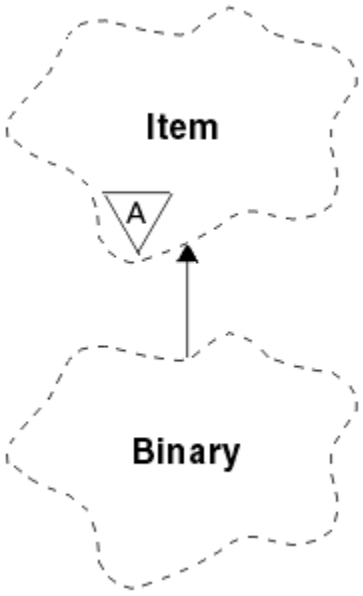


Abbildung 47. ImqBinary, Klasse

Objektattribute

data
Eine Bytefeldgruppe aus Binärdaten. Der Anfangswert ist null.

data length
Die Bytezahl. Der Anfangswert ist null.

data pointer
Die Adresse des ersten Byte der Daten (**data**). Der Anfangswert ist null.

Konstruktoren

ImqBinary( );
Der Standardkonstruktor.

ImqBinary(const ImqBinary & *binär*);
Der Kopierkonstruktor.

ImqBinary(const void * *data*, const size_t *length*);
Kopiert *length* Bytes aus *data*.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Kopiert die Daten (**data**) in den Nachrichtenpuffer und ersetzt dabei alle bereits vorhandenen Inhalte. Setzt das *msg* **-Format** auf MQFMT_NONE.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqItem".

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Setzt die Daten (**data**) durch Übertragen der verbleibenden Daten aus dem Nachrichtenpuffer und ersetzt dabei die bereits vorhandenen Daten (**data**).

Um erfolgreich zu sein, muss die ImqMessage **Format** MQFMT_NONE sein.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqItem".

Objektmethoden (öffentlich)

void operator = (const ImqBinary & *binary*);
Kopiert Bytes aus *binary*.

ImqBoolean Operator == (const ImqBinary & *binär* ;
Vergleicht dieses Objekt mit *binary*. Bei Ungleichheit wird FALSE zurückgegeben, andernfalls TRUE. Die Objekte sind gleich, wenn sie dieselbe Datenlänge (**data length**) aufweisen und die Bytezahl übereinstimmt.

ImqBoolean copyOut(void * *buffer*, const size_t *length*, const char *pad* = 0);
Kopiert eine Bytezahl bis zum Wert für *length* aus dem Datenzeiger (**data pointer**) in den Puffer (*buffer*). Wenn der Wert für die Datenlänge (**data length**) nicht ausreichend ist, wird der verbleibende Speicherplatz im Puffer (*buffer*) mit der für *pad* angegebenen Anzahl Bytes aufgefüllt. Der *Puffer* darf den Wert null aufweisen, wenn die *Länge* ebenfalls null beträgt. *length* darf nicht negativ sein. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

size_t dataLength( ) const ;
Gibt die Datenlänge (**data length**) zurück.

ImqBoolean setDataLength(const size_t *length*);
Legt die Datenlänge (**data length**) fest. Wenn **data length** infolge dieser Methode geändert wird, sind die Daten in diesem Objekt nicht initialisiert. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

void * dataPointer( ) const ;
Gibt den Datenzeiger (**data pointer**) zurück.

ImqBoolean isNull( ) const ;
TRUE wird zurückgegeben, wenn die Datenlänge (**data length**) null ist oder wenn alle Bytes für **data** null sind. Andernfalls wird FALSE zurückgegeben.

ImqBoolean set(const void * *buffer*, const size_t *length*);
Kopiert *length* Bytes aus *buffer*. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Objektmethoden (geschützt)

void clear( );
Reduziert die Datenlänge (**data length**) bis auf null.

Ursachencodes

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

C++-Klasse "ImqCache"

Verwenden Sie diese Klasse, um Daten im Speicher zu halten oder anzuordnen.

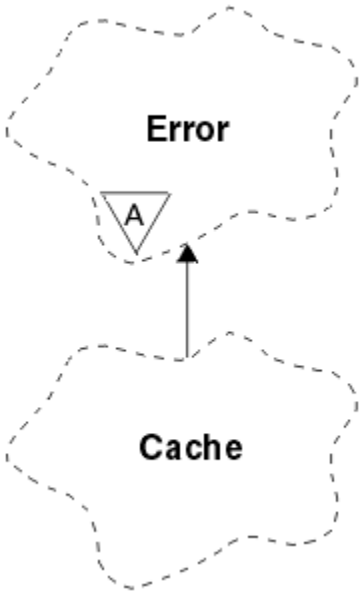


Abbildung 48. ImqCache, Klasse

Verwenden Sie diese Klasse, um Daten im Speicher zu halten oder anzuordnen. Sie können einen Pufferspeicher mit fester Größe angeben oder das System kann automatisch eine flexible Speicherkapazität bereitstellen. Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqCache" auf Seite 1346 aufgelisteten MQI-Aufrufe.

Objektattribute

automatic buffer
Gibt an, ob Pufferspeicher automatisch durch das System verwaltet (TRUE) oder vom Benutzer bereitgestellt (FALSE) wird. Die anfängliche Einstellung ist TRUE.

Dieses Attribut wird nicht direkt gesetzt. Es wird indirekt mithilfe der Methode **useEmptyBuffer** oder der Methode **useFullBuffer** festgelegt.

Wenn Speicher vom Benutzer bereitgestellt wird, ist die Einstellung für dieses Attribut FALSE; der Pufferspeicher kann sich nicht vergrößern, sodass Fehler durch Pufferüberlauf auftreten können. Adresse und Größe des Puffers bleiben konstant.

Wird kein Benutzerspeicher bereitgestellt, ist die Einstellung für dieses Attribut TRUE und der Pufferspeicher kann sich schrittweise vergrößern, um eine beliebige Nachrichtendatenmenge aufzunehmen. Durch die Vergrößerung des Puffers kann sich jedoch die Adresse des Puffers ändern, was bei der Verwendung des Pufferzeigers (**buffer pointer**) und des Datenzeigers (**data pointer**) unbedingt zu beachten ist.

buffer length
Die Größe des Pufferspeichers in Byte. Der Anfangswert ist null.

buffer pointer
Die Adresse des Pufferspeichers. Der Anfangswert ist null.

data length
Die Byteanzahl, die nach dem Datenzeiger (**data pointer**) folgt. Diese muss gleich dem oder kleiner als der Wert für die Nachrichtenlänge (**message length**) sein. Der Anfangswert ist null.

data offset
Die Byteanzahl vor dem Datenzeiger (**data pointer**). Diese muss gleich dem oder kleiner als der Wert für die Nachrichtenlänge (**message length**) sein. Der Anfangswert ist null.

data pointer
Die Adresse des Teils des Puffers, in den als Nächstes geschrieben oder aus dem als Nächstes ausgelesen werden soll. Der Anfangswert ist null.

message length
Die Byteanzahl der signifikanten Daten im Puffer. Der Anfangswert ist null.

Konstruktoren

ImqCache( );
Der Standardkonstruktor.

ImqCache(const ImqCache & ***Cache*** **);**
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqCache & ***Cache*** **);**
Kopiert Daten mit einer Bytezahl bis zur Nachrichtenlänge (**message length**) aus dem *cache*-Objekt in das Objekt. Wenn **automatic buffer** auf FALSE gesetzt ist, muss die Puffergröße (**buffer length**) bereits ausreichen, um die kopierten Daten aufzunehmen.

ImqBoolean automaticBuffer( ) const ;
Gibt den Wert für den automatischen Puffer (**automatic buffer**) zurück.

size_t bufferLength( ) const ;
Gibt die Puffergröße (**buffer length**) zurück.

char * bufferPointer( ) const ;
Gibt den Pufferzeiger (**buffer pointer**) zurück.

void clearMessage( );
Setzt die Nachrichtenlänge (**message length**) und die relative Datenadresse (**data offset**) auf null.

size_t dataLength( ) const ;
Gibt die Datenlänge (**data length**) zurück.

size_t dataOffset( ) const ;
Gibt die relative Datenadresse (**data offset**) zurück.

ImqBoolean setDataOffset(const size_t ***offset*** **);**
Legt die relative Datenadresse (**data offset**) fest. Der Wert für die Nachrichtenlänge (**message length**) wird ggf. erhöht, um sicherzustellen, dass er nicht kleiner ist als **data offset**. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

char * dataPointer( ) const ;
Gibt eine Kopie des Datenzeigers (**data pointer**) zurück.

size_t messageLength( ) const ;
Gibt die Nachrichtenlänge (**message length**) zurück.

ImqBoolean setMessageLength(const size_t ***length*** **);**
Legt die Nachrichtenlänge (**message length**) fest. Erhöht ggf. die Puffergröße (**buffer length**), um sicherzustellen, dass der Wert für die Nachrichtenlänge (**message length**) nicht größer ist als **buffer length**. Verkleinert ggf. den Wert für die relative Datenadresse (**data offset**), um sicherzustellen, dass er nicht größer ist als **message length**. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean moreBytes(const size_t ***bytes-required*** **);**
Stellt sicher, dass gemäß *bytes-required* mehr Bytes (zum Schreiben) zwischen dem Datenzeiger (**data pointer**) und dem Ende des Puffers verfügbar sind. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Wenn **automatic buffer** auf TRUE gesetzt ist, wird, wie erforderlich, mehr Speicherplatz angefordert; andernfalls muss bereits die Puffergröße (**buffer length**) ausreichend sein.

ImqBoolean read(const size_t *länge*, char * & *externem-puffer*);
Kopiert *length* Bytes aus dem Puffer in *external-buffer*, beginnend an der Position des Datenzeigers (**data pointer**). Nach dem Kopieren der Daten wird der Wert für die relative Datenadresse (**data offset**) um den Wert für *length* erhöht. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean resizeBuffer(const size_t *length*);
Verändert die Puffergröße (**buffer length**), sofern **automatic buffer** auf TRUE gesetzt ist. Dies wird durch erneutes Zuordnen des Pufferspeichers erreicht. Es werden Daten mit einer Bytezahl bis zur Nachrichtenlänge (**message length**) aus dem bereits vorhandenen in den neuen Puffer kopiert. Die maximale kopierte Byteanzahl ist gleich dem Wert für *length*. Der Pufferzeiger (**buffer pointer**) wird geändert. Die Nachrichtenlänge (**message length**) und die relative Datenadresse (**data offset**) werden so weit wie möglich in den Grenzen des neuen Puffers gehalten. Bei erfolgreicher Ausführung wird TRUE zurückgegeben; wenn **automatic buffer** auf FALSE gesetzt ist, wird FALSE zurückgegeben.

Anmerkung: Diese Methode kann mit Ursachencode MQRC_STORAGE_NOT_AVAILABLE fehlschlagen, wenn Fehler im Zusammenhang mit den Systemressourcen vorliegen.

ImqBoolean useEmptyBuffer(const char * *external-buffer*, const size_t *length*);
Gibt einen leeren Benutzerpuffer an, wobei folgende Einstellungen vorgenommen werden: der Pufferzeiger (**buffer pointer**) verweist auf *external-buffer*, die Puffergröße (**buffer length**) auf *length* und die Nachrichtenlänge (**message length**) auf null. Führt die Methode **clearMessage** aus. Wenn der Puffer vollständig mit Daten gefüllt ist, verwenden Sie stattdessen die Methode **useFullBuffer**. Ist der Puffer nur teilweise mit Daten gefüllt, geben Sie den richtigen Umfang mithilfe der Methode **setMessageLength** an. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Diese Methode kann verwendet werden, um eine feste Speicherkapazität anzugeben, wie zuvor beschrieben (*external-buffer* ist nicht null und *length* ist ungleich null), wobei **automatic buffer** auf FALSE gesetzt würde, oder sie kann verwendet werden, um auf einen systemverwalteten flexiblen Speicher zurückzugreifen (*external-buffer* ist null und *length* ist null), wobei **automatic buffer** auf TRUE gesetzt würde.

ImqBoolean useFullBuffer(const char * *externalBuffer*, const size_t *length*);
Wie bei **useEmptyBuffer**, außer dass die Nachrichtenlänge (**message length**) auf *length* gesetzt ist. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean write(const size_t *length*, const char * *external-buffer*);
Kopiert *length* Bytes aus *external-buffer* in den Puffer, beginnend an der Position des Datenzeigers (**data pointer**). Nach dem Kopieren der Daten wird der Wert für die relative Datenadresse (**data offset**) um den Wert für *length* erhöht und der Wert für die Nachrichtenlänge (**message length**) wird ggf. erhöht, um sicherzustellen, dass er nicht kleiner ist als der neue Wert für **data offset**. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Wenn **automatic buffer** auf TRUE gesetzt ist, wird eine hinreichende Speicherkapazität garantiert; andernfalls darf der endgültige Wert für die relative Datenadresse (**data offset**) den Wert für die Puffergröße (**buffer length**) nicht überschreiten.

Ursachencodes

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_TRUNCATED
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_ZERO_LENGTH

C++-Klasse "ImqChannel"

Diese Klasse bindet eine Kanaldefinition (MQCD) ein, die bei Ausführung der Methode "Manager::connect" für benutzerdefinierte Clientverbindungen verwendet werden soll.

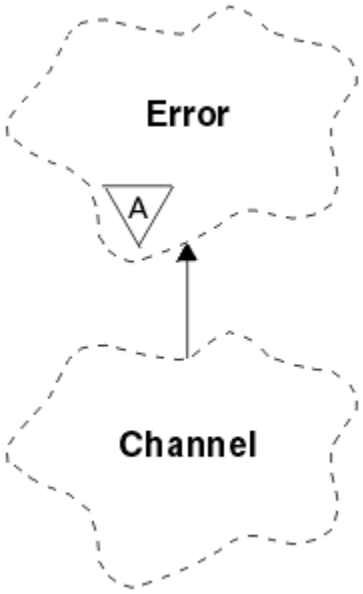


Abbildung 49. ImqChannel, Klasse

Weitere Informationen finden Sie in der Beschreibung der Methode "Manager: :connect" und im Abschnitt Beispielprogramm HELLO WORLD (imqwrld.cpp) . Nicht alle der aufgelisteten Methoden sind auf alle Plattformen anwendbar; weitere Informationen finden Sie in den Beschreibungen der Befehle DEFINE CHANNEL und ALTER CHANNEL in WebSphere MQ-Scriptbefehlsreferenz (MQSC). Die Klasse "ImqChannel" wird unter z/OS nicht unterstützt.

- „Objektattribute” auf Seite 1367
- „Konstruktoren” auf Seite 1368
- „Objektmethoden (öffentlich)” auf Seite 1369
- „Ursachencodes” auf Seite 1372

Objektattribute

batch heart-beat
Die Anzahl Millisekunden zwischen den Überprüfungen der Aktivität eines fernen Kanals. Der Anfangswert ist 0.

channel name
Der Name des Kanals. Der Anfangswert ist null.

connection name
Der Name der Verbindung. Beispielsweise die IP-Adresse eines Host-Computers. Der Anfangswert ist null.

Headerkomprimierung
Die Liste der vom Kanal unterstützten Komprimierungstechniken für Headerdaten. Die Anfangswerte sind alle auf MQCOMPRESS_NOT_AVAILABLE gesetzt.

heart-beat interval
Die Anzahl Sekunden zwischen den Überprüfungen, ob eine Verbindung noch besteht. Der Anfangswert ist 300.

keep alive interval
Die an den Kommunikationsstack übermittelte Anzahl Sekunden, die das Keepalive-Timing für den Kanal angibt. Der Anfangswert ist MQKAI_AUTO.

local address
Die lokale Datenübertragungsadresse für den Kanal.

maximum message length
Die maximale vom Kanal unterstützte Nachrichtenlänge innerhalb einer Kommunikation. Der Anfangswert ist 4 194 304.

message compression
Die Liste der vom Kanal unterstützten Komprimierungstechniken für Nachrichtendaten. Die Anfangswerte sind alle auf MQCOMPRESS_NOT_AVAILABLE gesetzt.

mode name
Der Name des Modus. Der Anfangswert ist null.

password
Ein für die Verbindungsauthentifizierung bereitgestelltes Kennwort. Der Anfangswert ist null.

receive exit count
Die Anzahl der Empfangsexits. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

receive exit names
Die Namen der Empfangsexits.

receive user data
Den Empfangsexits zugeordnete Daten.

security exit name
Der Name eines Sicherheitsexits, der auf der Serverseite der Verbindung aufgerufen werden soll. Der Anfangswert ist null.

security user data
Daten, die an den Sicherheitsexit übergeben werden sollen. Der Anfangswert ist null.

send exit count
Die Anzahl der Sendeexits. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

send exit names
Die Namen der Sendeexits.

send user data
Den Sendeexits zugeordnete Daten.

SSL CipherSpec
CipherSpec zur Verwendung mit SSL.

SSL client authentication type
Clientauthentifizierungstyp zur Verwendung mit SSL.

SSL peer name
Peername zur Verwendung mit SSL.

transaction program name
Der Name des Transaktionsprogramms. Der Anfangswert ist null.

transport type
Der Transporttyp der Verbindung. Der Anfangswert ist MQXPT_LU62.

user id
Eine für die Berechtigung bereitgestellte Benutzer-ID. Der Anfangswert ist null.

Konstruktoren

ImqChannel( ) ;
Der Standardkonstruktor.

ImqChannel(const ImqChannel & *Kanal*);
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqChannel & *Kanal*);
Kopiert Instanzdaten aus *channel* und ersetzt dabei alle bereits vorhandenen Instanzdaten.

MQLONG batchHeartBeat( ) const ;
Gibt das Überwachungssignal für den Stapel (**batch heart-beat**) zurück.

ImqBoolean setBatchHeartBeat(const MQLONG *heartbeat* = 0L);
Legt das Überwachungssignal für den Stapel (**batch heart-beat**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString channelName( ) const ;
Gibt den Kanalnamen (**channel name**) zurück.

ImqBoolean setChannelName(const char * *name* = 0);
Legt den Kanalnamen (**channel name**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString connectionName( ) const ;
Gibt den Verbindungsnamen (**connection name**) zurück.

ImqBoolean setConnectionName(const char * *name* = 0);
Legt den Verbindungsnamen (**connection name**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

size_t headerCompressionCount( ) const ;
Gibt die Anzahl der unterstützten Komprimierungstechniken für Headerdaten zurück.

ImqBoolean headerCompression(const size_t count, MQLONG compress [ ]) const ;
Gibt Kopien der unterstützten Komprimierungstechniken für Headerdaten in **compress** zurück. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setHeaderCompression(const size_t count, const MQLONG compress [ ]);

Setzt die unterstützten Komprimierungstechniken für Headerdaten auf **compress**.

Setzt die Anzahl der unterstützten Komprimierungstechniken für Headerdaten auf **count**.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG heartBeatInterval( ) const ;
Gibt das Intervall der Überwachungssignale (**heart-beat interval**) zurück.

ImqBoolean setHeartBeatInterval(const MQLONG *interval* = 300L);
Legt das Intervall der Überwachungssignale (**heart-beat interval**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG keepAliveInterval( ) const ;
Gibt das **Alive-Intervall** zurück.

ImqBoolean setKeepAliveInterval(const MQLONG *interval* = MQKAI_AUTO);
Legt das **Alive-Intervall** fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString localAddress() const ;
Gibt die lokale Adresse (**local address**) zurück.

ImqBoolean setLocalAddress (const char * *address* = 0);
Legt die lokale Adresse (**local address**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG maximumMessageLength( ) const ;
Gibt die maximale Nachrichtenlänge (**maximum message length**) zurück.

ImqBoolean setMaximumMessageLength(const MQLONG *length* = 4194304L);
Legt die maximale Nachrichtenlänge (**maximum message length**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

size_t messageCompressionCount( ) const ;
Gibt die Anzahl der unterstützten Komprimierungstechniken für Nachrichtendaten zurück.

ImqBoolean messageCompression(const size_t count, MQLONG compress [ ]) const ;
Gibt Kopien der unterstützten Komprimierungstechniken für Nachrichtendaten in **compress** zurück. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setMessageCompression(const size_t count, const MQLONG compress [ ]);

Setzt die unterstützten Komprimierungstechniken für Nachrichtendaten auf compress.

Setzt die Anzahl der unterstützten Komprimierungstechniken für Nachrichtendaten auf count.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString modeName( ) const ;
Gibt den Modusnamen (**mode name**) zurück.

ImqBoolean setModeName(const char * *name* = 0);
Legt den Modusnamen (**mode name**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString password( ) const ;
Gibt das Kennwort (**password**) zurück.

ImqBoolean setPassword(const char * *password* = 0);
Legt das Kennwort (**password**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

size_t receiveExitCount( ) const ;
Gibt die Anzahl der Empfangsexits (**receive exit count**) zurück.

ImqString receiveExitName( );
Gibt den ersten der Empfangsexitnamen (**receive exit names**) zurück, wenn vorhanden. Wenn die Anzahl der Empfangsexits (**receive exit count**) null ist, wird eine leere Zeichenfolge zurückgegeben.

ImqBoolean receiveExitNames(const size_t *count*, ImqString * *names* [ ]);
Gibt Kopien der Empfangsexitnamen (**receive exit names**) in *names* zurück. Setzt alle über die Anzahl der Empfangsexits (**receive exit count**) hinausgehenden Werte für *names* auf Nullzeichenfolgen. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveExitName(const char * *name* = 0);
Setzt alle Empfangsexitnamen (**receive exit names**) auf denselben Wert für *name*. *name* kann leer oder null sein. Setzt die Anzahl der Empfangsexits (**receive exit count**) auf 1 oder null. Löscht die Empfangsbenutzerdaten (**receive user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveExitNames(const size_t *count*, const char * *names* [ ]);
Setzt die Empfangsexitnamen (**receive exit names**) auf *names*. Die einzelnen Werte für *names* dürfen nicht leer oder null sein. Setzt die Anzahl der Empfangsexits (**receive exit count**) auf *count*. Löscht die Empfangsbenutzerdaten (**receive user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveExitNames(const size_t *count*, const ImqString * *names* [ ]);
Setzt die Empfangsexitnamen (**receive exit names**) auf *names*. Die einzelnen Werte für *names* dürfen nicht leer oder null sein. Setzt die Anzahl der Empfangsexits (**receive exit count**) auf *count*. Löscht die Empfangsbenutzerdaten (**receive user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString receiveUserData( );
Gibt den ersten Eintrag für Empfangsbenutzerdaten (**receive user data**) zurück, wenn vorhanden. Wenn die Anzahl der Empfangsexits (**receive exit count**) null ist, wird eine leere Zeichenfolge zurückgegeben.

ImqBoolean receiveUserData(const size_t *count*, ImqString * *data* [ ]);
Gibt Kopien der Einträge für Empfangsbenutzerdaten (**receive user data**) in *data* zurück. Setzt alle über die Anzahl der Empfangsexits (**receive exit count**) hinausgehenden Werte für *data* auf Nullzeichenfolgen. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveUserData(const char * *data* = 0);
Setzt die Empfangsbenutzerdaten (**receive user data**) auf denselben Wert für *data*. Wenn *data* nicht null ist, muss **Anzahl der Empfangsexits** mindestens 1 sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveUserData(const size_t *count*, const char * *data* [ ]);
Setzt die Empfangsbenutzerdaten (**receive user data**) auf *data*. *count* darf nicht größer als die Anzahl der Empfangsexits (**receive exit count**) sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setReceiveUserData(const size_t *count*, const ImqString * *data* [ ]);
Setzt die Empfangsbenutzerdaten (**receive user data**) auf *data*. *count* darf nicht größer als die Anzahl der Empfangsexits (**receive exit count**) sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString securityExitName( ) const ;
Gibt den Sicherheitsexitnamen (**security exit name**) zurück.

ImqBoolean setSecurityExitName(const char * *name* = 0);
Legt den Sicherheitsexitnamen (**security exit name**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString securityUserData( ) const ;
Gibt die Sicherheitsbenutzerdaten (**security user data**) zurück.

ImqBoolean setSecurityUserData(const char * *data* = 0);
Legt die Sicherheitsbenutzerdaten (**security user data**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

size_t sendExitCount( ) const ;
Gibt die Anzahl der Sendeexits (**send exit count**) zurück.

ImqString sendExitName( );
Gibt den ersten der Sendeexitnamen (**send exit names**) zurück, wenn vorhanden. Gibt eine leere Zeichenfolge zurück, wenn die Anzahl der Sendeexits (**send exit count**) null ist.

ImqBoolean sendExitNames(const size_t *count*, ImqString * *names* [ ]);
Gibt Kopien der Sendeexitnamen (**send exit names**) in *names* zurück. Setzt alle über die Anzahl der Sendeexits (**send exit count**) hinausgehenden Werte für *names* auf Nullzeichenfolgen. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendExitName(const char * *name* = 0);
Setzt alle Sendeexitnamen (**send exit names**) auf denselben Wert für *name*. *name* kann leer oder null sein. Setzt die Anzahl der Sendeexits (**send exit count**) auf 1 oder null. Löscht die Sendebenutzerdaten (**send user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendExitNames(const size_t *count*, const char * *names* [ ]);
Setzt die Sendeexitnamen (**send exit names**) auf *names*. Die einzelnen Werte für *names* dürfen nicht leer oder null sein. Setzt die Anzahl der Sendeexits (**send exit count**) auf *count*. Löscht die Sendebenutzerdaten (**send user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendExitNames(const size_t *count*, const ImqString * *names* [ ]);
Setzt die Sendeexitnamen (**send exit names**) auf *names*. Die einzelnen Werte für *names* dürfen nicht leer oder null sein. Setzt die Anzahl der Sendeexits (**send exit count**) auf *count*. Löscht die Sendebenutzerdaten (**send user data**). Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString sendUserData( );
Gibt das erste der **send user data**-Elemente zurück, sofern vorhanden. Gibt eine leere Zeichenfolge zurück, wenn die Anzahl der Sendeexits (**send exit count**) null ist.

ImqBoolean sendUserData(const size_t *count*, ImqString * *data* [ ]);
Gibt Kopien der Einträge für Sendebenutzerdaten (**send user data**) in *data* zurück. Setzt alle über die Anzahl der Sendeexits (**send exit count**) hinausgehenden Werte für *data* auf Nullzeichenfolgen. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendUserData(const char * *data* = 0);
Setzt alle Sendebenutzerdaten (**send user data**) auf denselben Wert für *data*. Wenn *data* nicht null ist, muss **send exit count** mindestens 1 sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendUserData(const size_t *count*, const char * *data* [ ]);
Setzt die Sendebenutzerdaten (**send user data**) auf *data*. *count* darf nicht größer als die Anzahl der Sendeexits (**send exit count**) sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setSendUserData(const size_t *count*, const ImqString * *data* [ ]);
Setzt die Sendebenutzerdaten (**send user data**) auf *data*. *count* darf nicht größer als die Anzahl der Sendeexits (**send exit count**) sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString sslCipherSpecification( ) const ;
Gibt die SSL-Verschlüsselungsspezifikation zurück.

ImqBoolean setSslCipherSpecification(const char * *name* = 0);
Legt die SSL-Verschlüsselungsspezifikation fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG sslClientAuthentication( ) const ;
Gibt den SSL-Clientauthentifizierungstyp zurück.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);
Legtt den SSL-Clientauthentifizierungstyp fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString sslPeerName( ) const ;
Gibt den SSL-Peernamen zurück.

ImqBoolean setSslPeerName(const char * *name* = 0);
Legt den SSL-Peernamen fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString transactionProgramName( ) const ;
Gibt den Transaktionsprogrammnamen (**transaction program name**) zurück.

ImqBoolean setTransactionProgramName(const char * *name* = 0);
Legt den Transaktionsprogrammnamen (**transaction program name**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG transportType( ) const ;
Gibt den Transporttyp (**transport type**) zurück.

ImqBoolean setTransportType(const MQLONG *type* = MQXPT_LU62);
Legt den Transporttyp (**transport type**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString userId( ) const ;
Gibt die Benutzer-ID (**user id**) zurück.

ImqBoolean setUserId(const char * *id* = 0);
Legt die Benutzer-ID (**user id**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Ursachencodes

- MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

C++-Klasse "ImqCICSBridgeHeader"

Diese Klasse bindet bestimmte Funktionen der MQCIH-Datenstruktur ein.

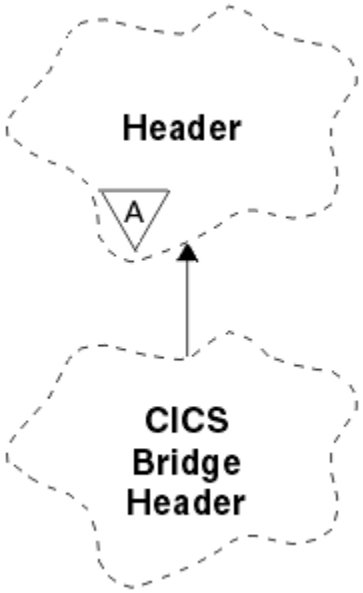


Abbildung 50. ImqCICSBridgeHeader, Klasse

Objekte dieser Klasse werden von Anwendungen verwendet, die Nachrichten über WebSphere MQ for z/OSan die CICS -Brücke senden.

- „Objektattribute” auf Seite 1373
- „Konstruktoren” auf Seite 1375
- „Methoden für überlastete ImqItem-Klassen” auf Seite 1376
- „Objektmethoden (öffentlich)” auf Seite 1376
- „Objektdaten (geschützt)” auf Seite 1378
- „Ursachencodes” auf Seite 1378
- „Rückgabecodes” auf Seite 1379

Objektattribute

ADS descriptor
ADS-Deskriptor zum Senden/Empfangen. Dieses Objektattribut wird mithilfe von MQCADSD_NONE gesetzt. Der Anfangswert ist MQCADSD_NONE. Folgende zusätzliche Werte sind möglich:

- MQCADSD_NONE
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

attention identifier
AID-Taste. Das Feld muss die Länge MQ_ATTENTION_ID_LENGTH aufweisen.

authenticator
RACF-Kennwort oder -Passticket. Der Anfangswert enthält Leerzeichen und weist die Länge MQ_AUTHENTICATOR_LENGTH auf.

bridge abend code
Bridge-Abbruchcode mit der Länge MQ_ABEND_CODE_LENGTH. Der Anfangswert besteht aus vier Leerzeichen. Der in diesem Feld zurückgegebene Wert ist abhängig vom Rückkehrcode. Weitere Informationen finden Sie in Tabelle 614 auf Seite 1379.

bridge cancel code
Transaktionscode für den Bridge-Abbruch. Das Feld ist reserviert und muss Leerzeichen sowie die Länge MQ_CANCEL_CODE_LENGTH aufweisen.

bridge completion code
Beendigungscode, der entweder den WebSphere MQ-Beendigungscode oder den CICS EIBRESP-Wert aufweisen kann. Der Anfangswert dieses Felds ist MQCC_OK. Der in diesem Feld zurückgegebene Wert ist abhängig vom Rückkehrcode. Weitere Informationen finden Sie in Tabelle 614 auf Seite 1379.

bridge error offset
Bridge-Fehler-Offset. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

bridge reason code
Der Ursachencode. Dieses Feld kann entweder die WebSphere MQ-Ursache oder den CICS EIBRESP2-Wert enthalten. Das Feld weist den Anfangswert MQRC_NONE auf. Der in diesem Feld zurückgegebene Wert ist abhängig vom Rückkehrcode. Weitere Informationen finden Sie in Tabelle 614 auf Seite 1379.

bridge return code
Rückkehrcode von der CICS-Brücke. Der Anfangswert ist MQCRC_OK.

conversational task
Gibt an, ob es sich um eine Dialogtask handeln kann. Der Anfangswert ist MQCCT_NO. Folgende zusätzliche Werte sind möglich:

- MQCCT_YES
- MQCCT_NO

cursor position
Cursorposition. Der Anfangswert ist null.

facility keep time
Freigabezeit der CICS-Brückenfunktion.

facility like
Vom Terminal emuliertes Attribut. Das Feld muss die Länge MQ_FACILITY_LIKE_LENGTH aufweisen.

facility token
BVT-Tokenwert. Das Feld muss die Länge MQ_FACILITY_LENGTH aufweisen. Der Anfangswert ist MQCFAC_NONE.

function
Funktion, die entweder den Namen des WebSphere MQ -Aufrufs oder die CICS -Funktion EIBFN enthalten kann Das Feld weist den Anfangswert MQCFUNC_NONE und die Länge MQ_FUNCTION_LENGTH auf. Der in diesem Feld zurückgegebene Wert ist abhängig vom Rückkehrcode. Weitere Informationen finden Sie unter Tabelle 614 auf Seite 1379.

Die folgenden zusätzlichen Werte sind möglich, wenn **function** einen WebSphere MQ-Aufrufnamen enthält:

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

get wait interval
Warteintervall für einen von der CICS-Brückentask ausgegebenen MQGET-Aufruf. Der Anfangswert ist MQCGWI_DEFAULT. Das Feld wird nur benötigt, wenn der Wert für **uow control** MQCUOWC_FIRST ist. Folgende zusätzliche Werte sind möglich:

- MQCGWI_DEFAULT

- MQWI_UNLIMITED

link type
Verbindungstyp. Der Anfangswert ist MQCLT_PROGRAM. Folgende zusätzliche Werte sind möglich:

- MQCLT_PROGRAM
- MQCLT_TRANSACTION

next transaction identifier
ID der nächsten anzuhängenden Transaktion. Das Feld muss die Länge MQ_TRANSACTION_ID_LENGTH aufweisen.

output data length
Datenlänge des Kommunikationsbereichs. Der Anfangswert ist MQCODL_AS_INPUT.

reply-to format
Formatname der Antwortnachricht. Der Anfangswert ist MQFMT_NONE und weist die Länge MQ_FORMAT_LENGTH auf.

start code
Startcode der Transaktion. Das Feld muss die Länge MQ_START_CODE_LENGTH aufweisen. Der Anfangswert ist MQCSC_NONE. Folgende zusätzliche Werte sind möglich:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

task end status
Taskendestatus. Der Anfangswert ist MQCTES_NOSYNC. Folgende zusätzliche Werte sind möglich:

- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

transaction identifier
ID der anzuhängenden Transaktion. Der Anfangswert muss Leerzeichen enthalten und die Länge MQ_TRANSACTION_ID_LENGTH aufweisen. Das Feld wird nur benötigt, wenn der Wert für **uow control** MQCUOWC_FIRST oder MQCUOWC_ONLY ist.

UOW control
UOW-Steuerung. Der Anfangswert ist MQCUOWC_ONLY. Folgende zusätzliche Werte sind möglich:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

Version
Die MQCIH-Versionsnummer. Der Anfangswert ist MQCIH_VERSION_2. Der einzige darüber hinaus unterstützte Wert ist MQCIH_VERSION_1.

Konstruktoren

ImqCICSBridgeHeader( );
Der Standardkonstruktor.

ImqCICSBridgeHeader (const ImqCICSBridgeHeader & *Header*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtuelle ImqBoolean copyOut(ImqMessage & *msg*);
Fügt eine MQCIH-Datenstruktur am Beginn des Nachrichtenpuffers ein, wobei bereits vorhandene Nachrichtendaten weiterverschoben werden, und setzt das Nachrichtenformat auf MQFMT_CICS.

Weitere Details finden Sie in der Beschreibung der übergeordneten Klassenmethoden.

virtuelle ImqBoolean pasteIn(ImqMessage & *Nachricht*);
Liest eine MQCIH-Datenstruktur aus dem Nachrichtenpuffer. Damit dieser Vorgang erfolgreich ist, muss als Codierung des *msg*-Objekts MQENC_NATIVE verwendet werden. Abrufen von Nachrichten mit MQGMO_CONVERT nach MQENC_NATIVE. Für eine erfolgreiche Ausführung muss das Format von "ImqMessage" MQFMT_CICS sein.

Weitere Details finden Sie in der Beschreibung der übergeordneten Klassenmethoden.

Objektmethoden (öffentlich)

void operator = (const ImqCICSBridgeHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

MQLONG ADSDescriptor( ) const;
Gibt eine Kopie des ADS-Deskriptors (**ADS descriptor**) zurück.

void setADSDescriptor(const MQLONG *descriptor* = MQCADSD_NONE);
Legt den ADS-Deskriptor (**ADS descriptor**) fest.

ImqString attentionIdentifier( ) const;
Gibt eine Kopie des AID-Zeichens (**attention identifier**) zurück, das bis zur Länge MQ_ATTENTION_ID_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setAttentionIdentifier(const char * *data* = 0);
Legt das AID-Zeichen (**attention identifier**) fest, das bis zur Länge MQ_ATTENTION_ID_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *data* bereitgestellt, wird **attention identifier** auf den Anfangswert zurückgesetzt.

ImqString authenticator( ) const;
Gibt eine Kopie des Authentifikators (**authenticator**) zurück, der bis zur Länge MQ_AUTHENTICATOR_LENGTH mit Leerzeichen aufgefüllt ist.

void setAuthenticator(const char * *data* = 0);
Legt den Authentifikator (**authenticator**) fest, der bis zur Länge MQ_AUTHENTICATOR_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *data* bereitgestellt, wird **authenticator** auf den Anfangswert zurückgesetzt.

ImqString bridgeAbendCode( ) const;
Gibt eine Kopie des Bridge-Abbruchcodes (**bridge abend**) zurück, der bis zur Länge MQ_ABEND_CODE_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

ImqString bridgeCancelCode( ) const;
Gibt eine Kopie des Bridge-Abbruchcodes (**bridge cancel code**) zurück, der bis zur Länge MQ_CANCEL_CODE_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setBridgeCancelCode(const char * *data* = 0);
Legt den Bridge-Abbruchcode (**bridge cancel code**) fest, der bis zur Länge MQ_CANCEL_CODE_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *data* bereitgestellt, wird **bridge cancel code** auf den Anfangswert zurückgesetzt.

MQLONG bridgeCompletionCode( ) const;
Gibt eine Kopie des Bridge-Beendigungscodes (**bridge completion code**) zurück.

MQLONG bridgeErrorOffset( ) const ;
Gibt eine Kopie des Bridge-Fehler-Offsets (**bridge error offset**) zurück.

MQLONG bridgeReasonCode( ) const;
Gibt eine Kopie des Bridge-Ursachencodes (**bridge reason code**) zurück.

MQLONG bridgeReturnCode( ) const;
Gibt den Bridge-Rückkehrcode (**bridge return code**) zurück.

MQLONG conversationalTask( ) const;
Gibt eine Kopie der Dialogtask (**conversational task**) zurück.

void setConversationalTask(const MQLONG *task* = MQCCT_NO);
Legt die Dialogtask (**conversational task**) fest.

MQLONG cursorPosition( ) const ;
Gibt eine Kopie der Cursorposition (**cursor position**) zurück.

void setCursorPosition(const MQLONG *position* = 0);
Legt die Cursorposition (**cursor position**) fest.

MQLONG facilityKeepTime( ) const;
Gibt eine Kopie der Beibehaltungszeit der Funktion (**facility keep time**) zurück.

void setFacilityKeepTime(const MQLONG *time* = 0);
Legt die Beibehaltungszeit der Funktion (**facility keep time**) fest.

ImqString facilityLike( ) const;
Gibt eine Kopie der **Funktion wie** zurück, die mit abschließenden Leerzeichen aufgefüllt wurde, um die Länge MQ_FACILITY_LIKE_LENGTH zu erreichen.

void setFacilityLike(const char * *name* = 0);
Legt **Funktion wie** fest, die mit abschließenden Leerzeichen aufgefüllt wird, zur Länge MQ_FACILITY_LIKE_LENGTH. Wird kein Wert für *name* bereitgestellt, so wird **facility like** auf den Anfangswert zurückgesetzt.

ImqBinary facilityToken( ) const;
Gibt eine Kopie des Funktionstokens (**facility token**) zurück.

ImqBoolean setFacilityToken (const ImqBinary & *Token*);
Legt das Funktionstoken (**facility token**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_FACILITY_LENGTH sein. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

void setFacilityToken(const MQBYTE8 *token* = 0);
Legt das Funktionstoken (**facility token**) fest. Der Wert für *token* kann null sein, was der Angabe von MQCFAC_NONE entspricht. Wenn der Wert für *token* ungleich null ist, muss er MQ_FACILITY_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQCFAC_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen. Beispiel: (MQBYTE *)MQCFAC_NONE.

ImqString function( ) const;
Gibt eine Kopie der Funktion (**function**) zurück, die bis zur Länge MQ_FUNCTION_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

MQLONG getWaitInterval( ) const;
Gibt eine Kopie des Abrufwarteintervalls (**get wait interval**) zurück.

void setGetWaitInterval(const MQLONG *interval* = MQCGWI_DEFA
Legt das Abrufwarteintervall (**get wait interval**) fest.

MQLONG linkType( ) const;
Gibt eine Kopie des Verbindungstyps (**link type**) zurück.

void setLinkType(const MQLONG *type* = MQCLT_PROGRAM);
Legt den Verbindungstyp (**link type**) fest.

ImqString nextTransactionIdentifier( ) const ;
Gibt eine Kopie der Daten der nächsten Transaktions-ID (**next transaction identifier**) zurück, die bis zur Länge MQ_TRANSACTION_ID_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

MQLONG outputDataLength( ) const;
Gibt eine Kopie der Ausgabedatenlänge (**output data length**) zurück.

void setOutputDataLength(const MQLONG *length* = MQCODL_AS_INPUT);
Legt die Ausgabedatenlänge (**output data length**) fest.

ImqString replyToFormat( ) const;
Gibt eine Kopie des Antwortformats (**reply-to format**) zurück, die bis zur Länge MQ_FORMAT_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setReplyToFormat(const char * *name* = 0);
Legt das Antwortformat (**reply-to format**) fest, das bis zur Länge MQ_FORMAT_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *name* bereitgestellt, wird **reply-to format** auf den Anfangswert zurückgesetzt.

ImqString startCode( ) const;
Gibt eine Kopie des Startcodes (**start code**) zurück, der bis zur Länge MQ_START_CODE_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setStartCode(const char * *data* = 0);
Legt die Daten des Startcodes (**start code**) fest, der bis zur Länge MQ_START_CODE_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *data* bereitgestellt, wird **start code** auf den Anfangswert zurückgesetzt.

MQLONG taskEndStatus( ) const;
Gibt eine Kopie des Taskendestatus (**task end status**) zurück.

ImqString transactionIdentifier( ) const;
Gibt eine Kopie der Daten der Transaktions-ID (**transaction identifier**) zurück, die bis zur Länge MQ_TRANSACTION_ID_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setTransactionIdentifier(const char * *data* = 0);
Legt die Transaktions-ID (**transaction identifier**) fest, die bis zur Länge MQ_TRANSACTION_ID_LENGTH mit abschließenden Leerzeichen aufgefüllt ist. Wird kein Wert für *data* bereitgestellt, wird **transaction identifier** auf den Anfangswert zurückgesetzt.

MQLONG UOWControl( ) const;
Gibt eine Kopie der UOW-Steuerung (**UOW control**) zurück.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);
Legt die UOW-Steuerung (**UOW control**) fest.

MQLONG version( ) const;
Gibt die **Version**-Nummer zurück.

ImqBoolean setVersion(const MQLONG *version* = MQCIH_VERSION_2);
Legt die **Version**-Nummer fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Objektdaten (geschützt)

MQLONG *olVersion*
Die höchstmögliche MQCIH-Versionsnummer, für die der für *opcih* zugeordnete Speicher ausreichend ist.

PMQCIH *opcih*
Die Adresse einer MQCIH-Datenstruktur. Die Größe des zugeordneten Speichers wird durch *olVersion* angegeben.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_WRONG_VERSION

Rückgabecodes

Tabelle 614. Rückkehrcodes der Klasse "ImqCICSBridgeHeader"

Rückkehrcode	Funktion	CompCode	Ursache	Abbruch-code
MQCRC_OK				
MQCRC_BRIDGE_ERROR			MQFB_CICS	
MQCRC_MQ_API_ERROR	WebSphere MQ-Aufrufna-me	WebSphere MQ CompCode	WebSphere MQ-Ursache	
MQCRC_BRIDGE_TIMEOUT	WebSphere MQ-Aufrufna-me	WebSphere MQ CompCode	WebSphere MQ-Ursache	
MQCRC_CICS_EXEC_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_SECURITY_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_PROGRAM_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				CICS ABC-ODE
MQCRC_APPLICATION_ABEND				CICS ABC-ODE

C++-Klasse "ImqDeadLetterHeader"

Diese Klasse bindet Funktionen der MQDLH-Datenstruktur ein.

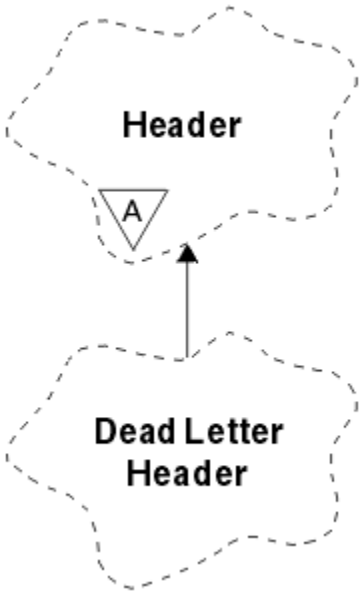


Abbildung 51. ImqDeadLetterHeader, Klasse

Objekte dieser Klasse werden typischerweise verwendet, wenn eine Anwendung auf eine Nachricht stößt, die nicht verarbeitet werden kann. Eine neue Nachricht mit einem Header für nicht zustellbare Nachrichten und der Nachrichteninhalt werden in die Warteschlange für nicht zustellbare Nachrichten eingereiht und die Nachricht wird gelöscht.

Objektattribute

dead-letter reason code
Die Ursache für die Aufnahme der Nachricht in die Warteschlange für nicht zustellbare Nachrichten. Der Anfangswert lautet MQRC_NONE.

destination queue manager name
Der Name des ursprünglichen Zielwarteschlangenmanagers. Der Name ist eine Zeichenfolge der Länge MQ_Q_MGR_NAME_LENGTH. Sein Anfangswert ist null.

destination queue name
Der Name der ursprünglichen Zielwarteschlange. Der Name ist eine Zeichenfolge der Länge MQ_Q_NAME_LENGTH. Sein Anfangswert ist null.

put application name
Der Name der Anwendung, von der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Der Name ist eine Zeichenfolge der Länge MQ_PUT_APPL_NAME_LENGTH. Sein Anfangswert ist null.

put application type
Der Typ der Anwendung, von der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Der Anfangswert ist null.

put date
Das Datum, an dem die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Das Datum ist eine Zeichenfolge der Länge MQ_PUT_DATE_LENGTH. Sein Anfangswert ist eine Nullzeichenfolge.

put time
Die Uhrzeit, zu der die Nachricht in die Warteschlange für nicht zustellbare Nachrichten eingereiht wurde. Die Uhrzeit ist eine Zeichenfolge der Länge MQ_PUT_TIME_LENGTH. Sein Anfangswert ist eine Nullzeichenfolge.

Konstruktoren

ImqDeadLetterHeader( );
Der Standardkonstruktor.

ImqDeadLetterHeader(const ImqDeadLetterHeader & *header*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Fügt eine MQDLH-Datenstruktur am Beginn des Nachrichtenpuffers ein, wobei bereits vorhandene Nachrichtendaten weiterverschoben werden. Setzt das *msg* **-Format** auf MQFMT_DEAD_LETTER_HEADER.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqHeader" auf Seite „C++-Klasse "ImqHeader"” auf Seite 1388.

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Liest eine MQDLH-Datenstruktur aus dem Nachrichtenpuffer.

Für eine erfolgreiche Ausführung muss ImqMessage **format** MQFMT_DEAD_LETTER_HEADER sein.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqHeader" auf Seite „C++-Klasse "ImqHeader"” auf Seite 1388.

Objektmethoden (öffentlich)

void operator = (const ImqDeadLetterHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

MQLONG deadLetterReasonCode( ) const ;
Gibt den Ursachencode für nicht zustellbare Nachrichten (**dead-letter reason code**) zurück.

void setDeadLetterReasonCode(const MQLONG *reason*);
Legt den Ursachencode für nicht zustellbare Nachrichten (**dead-letter reason code**) fest.

ImqString destinationQueueManagerName( ) const ;
Gibt den Namen des Zielwarteschlangenmanagers (**destination queue manager name**) zurück, von dem alle abschließenden Leerzeichen entfernt wurden.

void setDestinationQueueManagerName(const char * *name*);
Legt den Namen des Zielwarteschlangenmanagers (**destination queue manager name**) fest. Schneidet Zeichenfolgen ab, die länger sind als MQ_Q_MGR_NAME_LENGTH (48 Zeichen).

ImqString destinationQueueName( ) const ;
Gibt eine Kopie des Zielwarteschlangennamens (**destination queue name**) zurück, von dem alle abschließenden Leerzeichen entfernt wurden.

void setDestinationQueueName(const char * *name*);
Legt den Zielwarteschlangennamen (**destination queue name**) fest. Schneidet Zeichenfolgen ab, die länger sind als MQ_Q_NAME_LENGTH (48 Zeichen).

ImqString putApplicationName( ) const ;
Gibt eine Kopie des Namens der PUT-Anwendung (**put application name**) zurück, von dem alle abschließenden Leerzeichen entfernt wurden.

void setPutApplicationName(const char * *name* = 0);
Legt den Namen der PUT-Anwendung (**put application name**) fest. Schneidet Zeichenfolgen ab, die länger sind als MQ_PUT_APPL_NAME_LENGTH (28 Zeichen).

MQLONG putApplicationType( ) const ;
Gibt den Typ der PUT-Anwendung (**put application type**) zurück.

void setPutApplicationType(const MQLONG *type* = MQAT_NO_CONTEXT);
Legt den Typ der PUT-Anwendung (**put application type**) fest.

ImqString putDate( ) const ;
Gibt eine Kopie des PUT-Datums (**put date**) zurück, von dem alle abschließenden Leerzeichen entfernt wurden.

void setPutDate(const char * *date* = 0);
Legt das PUT-Datum (**put date**) fest. Schneidet Zeichenfolgen ab, die länger sind als MQ_PUT_DATE_LENGTH (8 Zeichen).

ImqString putTime( ) const ;
Gibt eine Kopie der PUT-Zeit (**put time**) zurück, von der alle abschließenden Leerzeichen entfernt wurden.

void setPutTime(const char * *time* = 0);
Legt die PUT-Zeit (**put time**) fest. Schneidet Zeichenfolgen ab, die länger sind als MQ_PUT_TIME_LENGTH (8 Zeichen).

Objektdaten (geschützt)

MQDLH *omqdlh*
Die MQDLH-Datenstruktur.

Ursachencodes

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_ERROR
- MQRC_ENCODING_ERROR

C++-Klasse "ImqDistributionList"

Diese Klasse bindet eine dynamische Verteilerliste ein, die auf mindestens eine Warteschlange verweist, um eine Nachricht oder mehrere Nachrichten an mehrere Zieladressen zu senden.

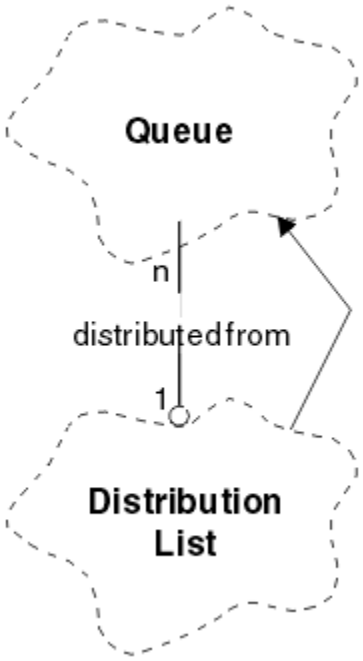


Abbildung 52. ImqDistributionList, Klasse

- „Objektattribute” auf Seite 1382
- „Konstruktoren” auf Seite 1382
- „Objektmethoden (öffentlich)” auf Seite 1383
- „Objektmethoden (geschützt)” auf Seite 1383

Objektattribute

first distributed queue
Das erste von mindestens einem Objekt der Klasse, wobei der Verteilerlistenverweis (**distribution list reference**) dieses Objekt nicht in einer bestimmten Reihenfolge anspricht.

Zu Beginn sind keine solchen Objekte vorhanden. Für das erfolgreiche Öffnen einer Klasse "ImqDistributionList" muss mindestens ein solches Objekt vorhanden sein.

Anmerkung: Wenn ein "ImqDistributionList"-Objekt geöffnet wird, werden alle geöffneten Objekte, die darauf verweisen, automatisch geschlossen.

Konstruktoren

ImqDistributionList( );
Der Standardkonstruktor.

ImqDistributionList (const ImqDistributionList & *Liste*);
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqDistributionList & *Liste*);
Bei allen Objekten, die auf **dieses** Objekt verweisen, wird der Verweis vor dem Kopieren entfernt. Nach dem Aufruf der Methode wird kein Objekt mehr auf **dieses** Objekt verweisen.

*** firstDistributedQueue( ) const ;**
Gibt die erste verteilte Warteschlange (**first distributed queue**) zurück.

Objektmethoden (geschützt)

void setFirstDistributedQueue(* *queue* = 0);
Legt die erste verteilte Warteschlange (**first distributed queue**) fest.

C++-Klasse "ImqError"

Diese abstrakte Klasse stellt Informationen zu den einem Objekt zugeordneten Fehlern bereit.

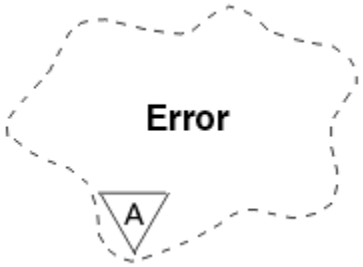


Abbildung 53. ImqError, Klasse

Objektattribute

completion code
Der aktuellste Beendigungscode. Der Anfangswert ist null. Folgende zusätzliche Werte sind möglich:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

reason code
Der aktuellste Ursachencode. Der Anfangswert ist null.

Konstruktoren

ImqError( );
Der Standardkonstruktor.

ImqError(const ImqError & *Fehler*);
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqError & *Fehler*);
Kopiert Instanzdaten aus *error* und ersetzt dabei die bereits vorhandenen Instanzdaten.

void clearErrorCodes( );
Setzt den Beendigungscode (**completion code**) und den Ursachencode (**reason code**) auf null.

MQLONG completionCode( ) const ;
Gibt den Beendigungscode (**completion code**) zurück.

MQLONG reasonCode( ) const ;
Gibt den Ursachencode (**reason code**) zurück.

Objektmethoden (geschützt)

ImqBoolean checkReadPointer(const void * *pointer*, const size_t *length*);
Prüft die Gültigkeit der Kombination aus Zeiger und Länge für den Lesezugriff und gibt, wenn die Kombination gültig ist, TRUE zurück.

ImqBoolean checkWritePointer(const void * *pointer*, const size_t *length*);
Prüft die Gültigkeit der Kombination aus Zeiger und Länge für den Schreib-/Lesezugriff und gibt, wenn die Kombination gültig ist, TRUE zurück.

void setCompletionCode(const MQLONG *code* = 0);
Legt den Beendigungscode (**completion code**) fest.

void setReasonCode(const MQLONG *code* = 0);
Legt den Ursachencode (**reason code**) fest.

Ursachencodes

- MQRC_BUFFER_ERROR

C++-Klasse "ImqGetMessageOptions"

Diese Klasse bindet die MQGMO-Datenstruktur ein.

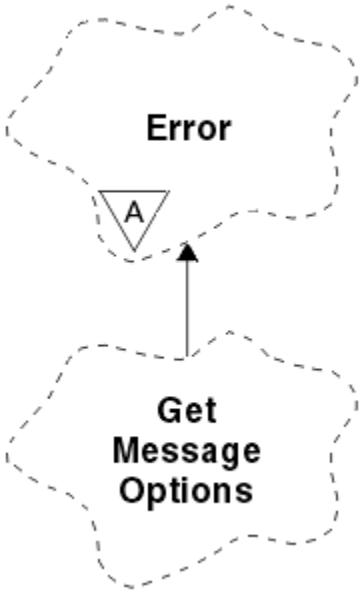


Abbildung 54. ImqGetMessageOptions, Klasse

Objektattribute

group status
Status einer Nachricht für eine Gruppe von Nachrichten. Der Anfangswert ist MQGS_NOT_IN_GROUP. Folgende zusätzliche Werte sind möglich:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

match options
Optionen für die Auswahl eingehender Nachrichten. Der Anfangswert ist MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Folgende zusätzliche Werte sind möglich:

- MQMO_GROUP_ID
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

Nachrichten-Token
Nachrichtentoken. Ein binärer Wert (MQBYTE16) mit der Länge MQ_MSG_TOKEN_LENGTH. Der Anfangswert lautet MQMTOK_NONE.

options
Optionen, die sich auf eine Nachricht beziehen. Der Anfangswert ist MQGMO_NO_WAIT. Folgende zusätzliche Werte sind möglich:

- 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

resolved queue name
Aufgelöster Warteschlangenname. Dieses Attribut ist schreibgeschützt. Namen sind nie länger als 48 Zeichen und können bis zu dieser Länge mit Nullen aufgefüllt werden. Der Anfangswert ist eine Nullzeichenfolge.

returned length
Zurückgegebene Länge. Der Anfangswert ist MQRL_UNDEFINED. Dieses Attribut ist schreibgeschützt.

segmentation
Die Möglichkeit zur Segmentierung einer Nachricht. Der Anfangswert ist MQSEG_INHIBITED. Der zusätzliche Wert MQSEG_ALLOWED ist möglich.

segment status
Der Segmentierungsstatus einer Nachricht. Der Anfangswert ist MQSS_NOT_A_SEGMENT. Folgende zusätzliche Werte sind möglich:

- MQSS_SEGMENT
- MQSS_LAST_SEGMENT

syncpoint participation
Auf TRUE gesetzt, wenn Nachrichten unter Synchronisationspunktsteuerung abgerufen werden.

wait interval
Die Zeit, während der die Klassenmethode **get** pausiert und auf eine passende eingehende Nachricht wartet, falls noch keine verfügbar ist. Der Anfangswert ist null, was zu einer unbegrenzten Wartezeit führt. Der zusätzliche Wert MQWI_UNLIMITED ist möglich. Dieses Attribut wird ignoriert, solange die Optionen (**options**) nicht MQGMO_WAIT einschließen.

Konstruktoren

ImqGetMessageOptions( );
Der Standardkonstruktor.

ImqGetMessageOptions(const ImqGetMessageOptions & *gmo*);
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqGetMessageOptions & *gmo*);
Kopiert Instanzdaten aus *gmo* und ersetzt dabei die bereits vorhandenen Instanzdaten.

MQCHAR groupStatus( ) const ;
Gibt den Gruppenstatus (**group status**) zurück.

void setGroupStatus(const MQCHAR *status*);
Legt den Gruppenstatus (**group status**) fest.

MQLONG matchOptions( ) const ;
Gibt die Abgleichoptionen (**match options**) zurück.

void setMatchOptions(const MQLONG *options*);
Legt die Abgleichoptionen (**match options**) fest.

ImqBinary messageToken( ) const;
Gibt das Nachrichtentoken (**message token**) zurück.

ImqBoolean setMessageToken (const ImqBinary & *Token*);
Legt das Nachrichtentoken (**message token**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_MSG_TOKEN_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setMessageToken(const MQBYTE16 *token* = 0);
Legt das Nachrichtentoken (**message token**) fest. *token* darf null sein, was einer Festlegung von MQMTOK_NONE entspricht. Wenn der Wert für *token* ungleich null ist, muss er MQ_MSG_TOKEN_LENGTH Bytes an binären Daten aufweisen.

Werden vordefinierte Werte wie beispielsweise MQMTOK_NONE verwendet, müssen Sie möglicherweise keine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQMTOK_NONE.

MQLONG options( ) const ;
Gibt die Optionen (**options**) zurück.

void setOptions(const MQLONG *options*);
Legt die Optionen (**options**) fest, einschließlich des Werts für die Synchronisationspunktbeteiligung (**syncpoint participation**).

ImqString resolvedQueueName( ) const ;
Gibt eine Kopie des aufgelösten Warteschlangennamens (**resolved queue name**) zurück.

MQLONG returnedLength( ) const;
Gibt die zurückgegebene Länge (**returned length**) zurück.

MQCHAR segmentation( ) const ;
Gibt die Segmentierung (**segmentation**) zurück.

void setSegmentation(const MQCHAR *value*);
Legt die Segmentierung (**segmentation**) fest.

MQCHAR segmentStatus( ) const ;
Gibt den Segmentstatus (**segment status**) zurück.

void setSegmentStatus(const MQCHAR *status*);
Legt den Segmentstatus (**segment status**) fest.

ImqBoolean syncPointParticipation( ) const ;
Gibt den Wert für die Synchronisationspunktbeteiligung (**syncpoint participation**) zurück, der auf TRUE gesetzt ist, wenn die Optionen (**options**) entweder MQGMO_SYNCPOINT oder MQGMO_SYNCPOINT_IF_PERSISTENT einschließen.

void setSyncPointParticipation(const ImqBoolean *sync*);
Setzt den Wert für die Synchronisationspunktbeteiligung (**syncpoint participation**). Wenn *sync* auf TRUE gesetzt ist, werden die Optionen (**options**) so geändert, dass sie MQGMO_SYNCPOINT einschließen und sowohl MQGMO_NO_SYNCPOINT als auch MQGMO_SYNCPOINT_IF_PERSISTENT ausschließen. Wenn *sync* auf FALSE gesetzt ist, werden die Optionen (**options**) so geändert, dass sie MQGMO_NO_SYNCPOINT einschließen und sowohl MQGMO_SYNCPOINT als auch MQGMO_SYNCPOINT_IF_PERSISTENT ausschließen.

MQLONG waitInterval( ) const ;
Gibt das Warteintervall (**wait alive interval**) zurück.

void setWaitInterval(const MQLONG *interval*);
Legt das Warteintervall (**wait interval**) fest.

Objektmethoden (geschützt)

static void setVersionSupported(const MQLONG);
Legt die **MQGMO**-Version fest. Nimmt standardmäßig den Wert **MQGMO_VERSION_3** an.

Objektdaten (geschützt)

MQGMO *omqgmo*
Eine Datenstruktur der Version 2 einer MQGMO. Zugriff auf MQGMO-Felder, die nur für MQGMO_VERSION_2 unterstützt werden.

PMQGMO *opgmo*
Die Adresse einer MQGMO-Datenstruktur. Die Versionsnummer für diese Adresse wird in *olVersion* angezeigt. Überprüfen Sie die Versionsnummer vor dem Zugriff auf MQGMO-Felder, um sicherzustellen, dass diese vorhanden sind.

MQLONG *olVersion*
Die Versionsnummer der MQGMO-Datenstruktur, die von *opgmo* adressiert wird.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR

C++-Klasse "ImqHeader"

Diese abstrakte Klasse bindet einheitliche Funktionen der MQDLH-Datenstruktur ein.

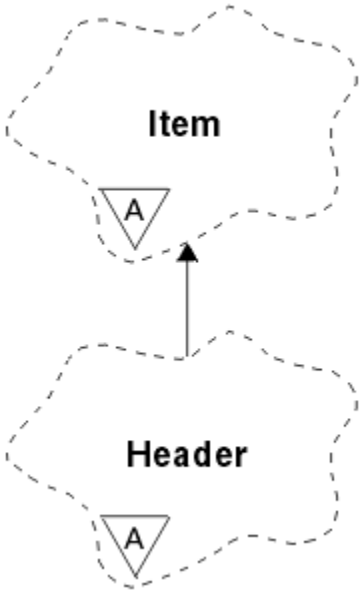


Abbildung 55. ImqHeader, Klasse

- „Objektattribute” auf Seite 1388
- „Konstruktoren” auf Seite 1388
- „Objektmethoden (öffentlich)” auf Seite 1389

Objektattribute

character set
Die ursprüngliche ID des codierten Zeichensatzes. Zu Beginn: MQCCSI_Q_MGR.

encoding
Die ursprüngliche Codierung. Zu Beginn: MQENC_NATIVE.

format
Das ursprüngliche Format. Zu Beginn: MQFMT_NONE.

header flags
Folgende Anfangswerte sind möglich:

- Null für Objekte der Klasse "ImqDeadLetterHeader"
- MQIIH_NONE für Objekte der Klasse "ImqIMSBridgeHeader"
- MQRMHF_LAST für Objekte der Klasse "ImqReferenceHeader"
- MQCIH_NONE für Objekte der Klasse "ImqCICSBridgeHeader"
- MQWIH_NONE für Objekte der Klasse "ImqWorkHeader"

Konstruktoren

ImqHeader( );
Der Standardkonstruktor.

ImqHeader(const ImqHeader & ***Header*** **);**
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

virtual MQLONG characterSet( ) const ;
Gibt den Zeichensatz (**character set**) zurück.

virtual void setCharacterSet(const MQLONG *ccsid* = MQCCSI_Q_MGR);
Legt den Zeichensatz (**character set**) fest.

virtual MQLONG encoding( ) const ;
Gibt die Codierung (**encoding**) zurück.

virtual void setEncoding(const MQLONG *encoding* = MQENC_NATIVE);
Legt die Codierung (**encoding**) fest.

virtual ImqString format( ) const ;
Gibt eine Kopie des Formats (**format**) zurück, einschließlich abschließender Leerzeichen.

virtual void setFormat(const char * *name* = 0);
Legt das **Format** fest, das auf 8 Zeichen mit abschließenden Leerzeichen aufgefüllt wird.

virtual MQLONG headerFlags( ) const ;
Gibt den **header flags**zurück.

virtual void setHeaderFlags(const MQLONG *flags* = 0);
Legt die **header flags** fest.

C++-Klasse "ImqIMSBridgeHeader"

Diese Klasse bindet Funktionen der MQIIH-Datenstruktur ein.

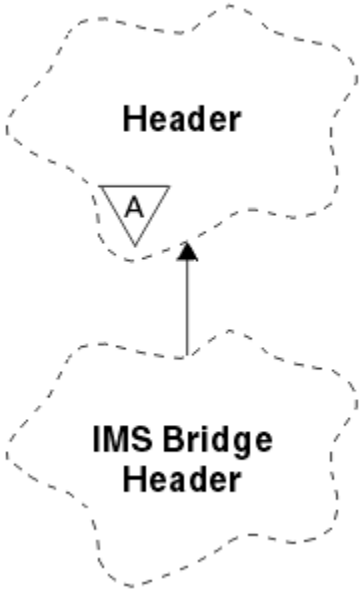


Abbildung 56. ImqIMSBridgeHeader, Klasse

Objekte dieser Klasse werden von Anwendungen verwendet, die Nachrichten an die IMS -Bridge über WebSphere MQ for z/OSsenden.

Anmerkung: Der Zeichensatz (**character set**) und die Codierung (**encoding**) von "ImqHeader" müssen Standardwerte aufweisen und dürfen nicht auf andere Werte gesetzt werden.

- „Objektattribute” auf Seite 1390
- „Konstruktoren” auf Seite 1390
- „Methoden für überlastete ImqItem-Klassen” auf Seite 1390

- „Objektmethoden (öffentlich)” auf Seite 1391
- „Objektdaten (geschützt)” auf Seite 1392
- „Ursachencodes” auf Seite 1392

Objektattribute

authenticator
RACF-Kennwort oder -Passticket mit der Länge MQ_AUTHENTICATOR_LENGTH. Der Anfangswert ist MQIAUT_NONE.

commit mode
Festschreibungsmodus. Weitere Informationen zu IMS -Commitmodi finden Sie im *OTMA-Benutzerhandbuch* . Der Anfangswert ist MQICM_COMMIT_THEN_SEND. Der zusätzliche Wert MQICM_SEND_THEN_COMMIT ist möglich.

logical terminal override
Überschreibung des logischen Terminals mit der Länge MQ_LTERM_OVERRIDE_LENGTH. Der Anfangswert ist eine Nullzeichenfolge.

message format services map name
MFS-Zuordnungsname mit der Länge MQ_MFS_MAP_NAME_LENGTH. Der Anfangswert ist eine Nullzeichenfolge.

reply-to format
Format einer beliebigen Antwort mit der Länge MQ_FORMAT_LENGTH. Der Anfangswert ist MQFMT_NONE.

security scope
Bereich der IMS-Sicherheitsverarbeitung. Der Anfangswert ist MQISS_CHECK. Der zusätzliche Wert MQISS_FULL ist möglich.

transaction instance id
Identität der Transaktionsinstanz, ein binärer Wert (MQBYTE16) mit der Länge MQ_TRAN_INSTANCE_ID_LENGTH. Der Anfangswert ist MQITII_NONE.

transaction state
Status des IMS-Dialogs. Der Anfangswert ist MQITS_NOT_IN_CONVERSATION. Der zusätzliche Wert MQITS_IN_CONVERSATION ist möglich.

Konstruktoren

ImqIMSBridgeHeader( );
Der Standardkonstruktor.

ImqIMSBridgeHeader (const ImqIMSBridgeHeader & *Header*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Fügt eine MQIIH-Datenstruktur am Beginn des Nachrichtenpuffers ein, wobei bereits vorhandene Nachrichtendaten weiterverschoben werden. Setzt das *msg* **-Format** auf MQFMT_IMS.

Weitere Informationen finden Sie in der Beschreibung der übergeordneten Klassenmethode.

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Liest eine MQIIH-Datenstruktur aus dem Nachrichtenpuffer.

Für eine erfolgreiche Ausführung muss die Codierung (**encoding**) des *msg*-Objekts MQENC_NATIVE sein. Abrufen von Nachrichten mit MQGMO_CONVERT nach MQENC_NATIVE.

Für eine erfolgreiche Ausführung muss das **Format** von ImqMessage MQFMT_IMS sein.

Weitere Informationen finden Sie in der Beschreibung der übergeordneten Klassenmethode.

Objektmethoden (öffentlich)

void operator = (const ImqIMSBridgeHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqString authenticator( ) const ;
Gibt eine Kopie des Authentifikators (**authenticator**) zurück, der bis zur Länge MQ_AUTHENTICATOR_LENGTH mit Leerzeichen aufgefüllt ist.

void setAuthenticator(const char * *name*);
Legt den Authentifikator (**authenticator**) fest.

MQCHAR commitMode( ) const ;
Gibt den Festschreibungsmodus (**commit mode**) zurück.

void setCommitMode(const MQCHAR *mode*);
Legt den Festschreibungsmodus (**commit mode**) fest.

ImqString logicalTerminalOverride( ) const ;
Gibt eine Kopie der Überschreibung des logischen Terminals (**logical terminal override**) zurück.

void setLogicalTerminalOverride(const char * *override*);
Legt die Überschreibung des logischen Terminals (**logical terminal override**) fest.

ImqString messageFormatServicesMapName( ) const ;
Gibt eine Kopie des Zuordnungsnamens für Nachrichtenformatservices (**message format services map name**) zurück.

void setMessageFormatServicesMapName(const char * *name*);
Legt den Zuordnungsnamen für Nachrichtenformatservices (**message format services map name**) fest.

ImqString replyToFormat( ) const ;
Gibt eine Kopie des Antwortformats (**reply-to format**) zurück, die bis zur Länge MQ_FORMAT_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

void setReplyToFormat(const char * *format*);
Legt das Antwortformat (**reply-to format**) fest, das bis zur Länge MQ_FORMAT_LENGTH mit abschließenden Leerzeichen aufgefüllt ist.

MQCHAR securityScope( ) const ;
Gibt den Sicherheitsbereich (**security scope**) zurück.

void setSecurityScope(const MQCHAR *scope*);
Legt den Sicherheitsbereich (**security scope**) fest.

ImqBinary transactionInstanceId( ) const ;
Gibt eine Kopie der Transaktionsinstanz-ID (**transaction instance id**) zurück.

ImqBoolean setTransactionInstanceId(const ImqBinary & *ID*);
Legt die Transaktionsinstanz-ID (**transaction instance id**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_INSTANCE_ID_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setTransactionInstanceId(const MQBYTE16 *id* = 0);
Legt die Transaktionsinstanz-ID (**transaction instance id**) fest. Der Wert für *id* kann null sein, was der Angabe von MQITII_NONE entspricht. Wenn der Wert für *id* ungleich null ist, muss er MQ_TRAN_INSTANCE_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQITII_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQITII_NONE.

MQCHAR transactionState( ) const ;
Gibt den Transaktionsstatus (**transaction state**) zurück.

void setTransactionState(const MQCHAR *state*);
Legt den Transaktionsstatus (**transaction state**) fest.

Objektdaten (geschützt)

MQIIH *omqiih*
Die MQIIH-Datenstruktur.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

C++-Klasse "ImqItem"

Diese abstrakte Klasse stellt ein Element, möglicherweise eines von mehreren, innerhalb einer Nachricht dar.

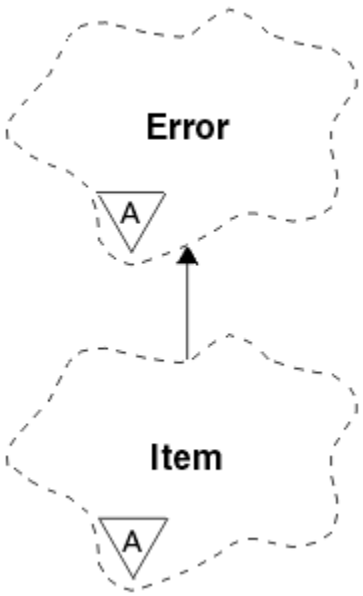


Abbildung 57. ImqItem, Klasse

Elemente werden in einem Nachrichtenpuffer miteinander verkettet. Jeder Elementtyp wird einer bestimmten Datenstruktur zugeordnet, die mit einer Struktur-ID beginnt.

In dieser abstrakten Klasse ermöglichen polymorphe Methoden das Kopieren von Elementen in und aus Nachrichten. Über die "ImqMessage"-Klassenmethoden **readItem** und **writeItem** können diese polymorphen Methoden in einer für Anwendungsprogramme natürlicheren Weise aufgerufen werden.

- „Objektattribute” auf Seite 1392
- „Konstruktoren” auf Seite 1393
- „Klassenmethoden (öffentlich)” auf Seite 1393
- „Objektmethoden (öffentlich)” auf Seite 1393
- „Ursachencodes” auf Seite 1393

Objektattribute

structure id
Eine Zeichenfolge aus vier Zeichen am Beginn einer Datenstruktur. Dieses Attribut ist schreibgeschützt. Das Attribut wird für abgeleitete Klassen empfohlen. Es ist nicht automatisch eingeschlossen.

Konstruktoren

ImqItem( );
Der Standardkonstruktor.

ImqItem(const ImqItem & *Element*);
Der Kopierkonstruktor.

Klassenmethoden (öffentlich)

static ImqBoolean structureIdIs(const char * *structure-id-to-test*, const ImqMessage & *msg*);
Gibt TRUE zurück, wenn die Struktur-ID (**structure id**) des nächsten "ImqItem" im eingehenden *msg* identisch mit *structure-id-to-test* ist. Das nächste Element wird als der Teil des Nachrichtenpuffers erkannt, der aktuell vom Datenzeiger (**data pointer**) von "ImqCache" angesprochen wird. Diese Methode stützt sich auf die Struktur-ID (**structure id**) und funktioniert daher nicht für alle abgeleiteten "ImqItem"-Klassen.

Objektmethoden (öffentlich)

void operator = (const ImqItem & *Element*);
Kopiert Instanzdaten aus *item* und ersetzt dabei die bereits vorhandenen Instanzdaten.

virtuelles ImqBoolean copyOut(ImqMessage & *msg*) = 0;
Schreibt dieses Objekt als nächstes Element in einen Puffer für abgehende Nachrichten und hängt es an bereits vorhandene Elemente an. Nach erfolgreicher Ausführung der Schreiboperation erhöht sich die Datenlänge (**data length**) von "ImqCache". Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Überschreiben Sie diese Methode, wenn Sie mit einer bestimmten Unterklasse arbeiten wollen.

virtuelle ImqBoolean pasteIn(ImqMessage & *msg*) = 0;
Löscht dieses Objekt beim Auslesen aus dem Puffer für eingehende Nachrichten. Das Auslesen erfolgt mit anschließendem Löschen, wobei der Datenzeiger (**data pointer**) von "ImqCache" weiterverschoben wird. Der Pufferinhalt bleibt jedoch gleich, sodass Daten durch Zurücksetzen des Datenzeigers (**data pointer**) von "ImqCache" erneut ausgelesen werden können.

Die (Unter-)Klasse dieses Objekts muss mit der Struktur-ID (**structure id**) konsistent sein, die im Nachrichtenpuffer des *msg*-Objekts als nächste gefunden wird.

Die Codierung (**encoding**) des *msg*-Objekts sollte MQENC_NATIVE sein. Es wird empfohlen, beim Abrufen von Nachrichten sicherzustellen, dass die Codierung (**encoding**) von "ImqMessage" auf MQENC_NATIVE gesetzt ist und die Optionen (**options**) von "ImqGetMessageOptions" MQGMO_CONVERT einschließen.

Nach erfolgreicher Ausführung der Leseoperation wird die Datenlänge (**data length**) von "ImqCache" reduziert. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Überschreiben Sie diese Methode, wenn Sie mit einer bestimmten Unterklasse arbeiten wollen.

Ursachencodes

- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA

C++-Klasse "ImqMessage"

Diese Klasse bindet eine MQMD-Datenstruktur ein und führt auch die Erstellung und Neuerstellung von Nachrichtendaten aus.

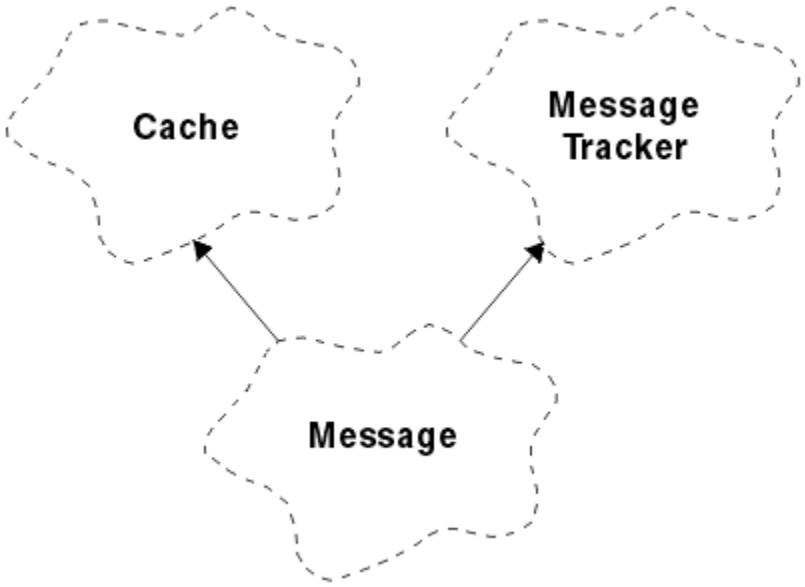


Abbildung 58. ImqMessage, Klasse

- „Objektattribute” auf Seite 1394
- „Konstruktoren” auf Seite 1398
- „Objektmethoden (öffentlich)” auf Seite 1398
- „Objektmethoden (geschützt)” auf Seite 1400
- „Objektdaten (geschützt)” auf Seite 1400

Objektattribute

application id data
Einer Nachricht zugeordnete Identitätsinformationen. Der Anfangswert ist eine Nullzeichenfolge.

application origin data
Ursprüngliche einer Nachricht zugeordnete Informationen. Der Anfangswert ist eine Nullzeichenfolge.

backout count
Die Anzahl der Male, die eine Nachricht nach einem Abrufversuch zurückgesetzt wurde. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

character set
ID des codierten Zeichensatzes. Der Anfangswert ist MQCCSI_Q_MGR. Folgende zusätzliche Werte sind möglich:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

Sie können auch eine selbst gewählte ID für den codierten Zeichensatz verwenden. Weitere Informationen hierzu finden Sie im Abschnitt „Codepagekonvertierung” auf Seite 941.

encoding
Die Maschinencodierung der Nachrichtendaten. Der Anfangswert ist MQENC_NATIVE.

expiry
Eine zeitabhängige Größe, die steuert, wie lange WebSphere MQ eine nicht abgerufene Nachricht beibehält, bevor sie gelöscht wird. Der Anfangswert ist MQEI_UNLIMITED.

format
Der Name des Formats (Vorlage) zur Beschreibung des Layouts von Daten im Puffer. Namen, die mehr als acht Zeichen aufweisen, werden auf acht Zeichen abgeschnitten. Namen werden immer mit Leerzeichen auf acht Zeichen aufgefüllt. Der Anfangskonstantenwert ist MQFMT_NONE. Folgende zusätzliche Konstanten sind möglich:

- 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

Sie können auch eine anwendungsspezifische Zeichenfolge Ihrer Wahl verwenden. Weitere Informationen hierzu finden Sie im Feld „Format (MQCHAR8)" auf Seite 414 des Nachrichtendeskriptors (MQMD).

message flags
Informationen zur Segmentierungssteuerung. Der Anfangswert ist MQMF_SEGMENTATION_INHIBITED. Folgende zusätzliche Werte sind möglich:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE

message type
Die breite Kategorisierung einer Nachricht. Der Anfangswert ist MQMT_DATAGRAM. Folgende zusätzliche Werte sind möglich:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_FIRST
- MQMT_APPL_LAST

Sie können auch einen anwendungsspezifischen Wert Ihrer Wahl verwenden. Weitere Informationen hierzu finden Sie im Feld „MsgType (MQLONG)" auf Seite 427 des Nachrichtendeskriptors (MQMD).

offset
Offset-Informationen. Der Anfangswert ist null.

original length
Die ursprüngliche Länge einer segmentierten Nachricht. Der Anfangswert ist MQOL_UNDEFINED.

persistence
Zeigt an, dass die Nachricht wichtig ist und mithilfe eines persistenten Speichers ununterbrochen gesichert werden muss. Diese Option bringt eine Leistungsbeeinträchtigung mit sich. Der Anfangswert ist MQPER_PERSISTENCE_AS_Q_DEF. Folgende zusätzliche Werte sind möglich:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority
Die relative Priorität für Übertragung und Zustellung. Nachrichten mit derselben Priorität werden in der Regel in der Reihenfolge ihrer Bereitstellung zugestellt (obwohl verschiedene Kriterien erfüllt sein müssen, um dies zu gewährleisten). Der Anfangswert ist MQPRI_PRIORITY_AS_Q_DEF.

property validation
Gibt an, ob eine Eigenschaftsüberprüfung ausgeführt werden soll, wenn eine Eigenschaft für die Nachricht gesetzt wird. Der Anfangswert ist **MQCMHO_DEFAULT_VALIDATION**. Folgende zusätzliche Werte sind möglich:

- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

Folgende Methoden wirken sich auf die Eigenschaftsüberprüfung (**property validation**) aus:

MQLONG propertyValidation( ) const ;
Gibt die Option der Eigenschaftsüberprüfung (**property validation**) zurück.

void setPropertyValidation(const MQLONG *option*);
Legt die Option der Eigenschaftsüberprüfung (**property validation**) fest.

put application name
Der Name der Anwendung, die eine Nachricht eingereiht hat. Der Anfangswert ist eine Nullzeichenfolge.

put application type
Der Typ der Anwendung, die eine Nachricht eingereiht hat. Der Anfangswert ist MQAT_NO_CONTEXT. Folgende zusätzliche Werte sind möglich:

- MQAT_AIX
- MQAT_CICS
- MQAT_CICS_BRIDGE
- MQAT_DOS
- MQAT_IMS
- MQAT_IMS_BRIDGE
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_QMGR
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT

- MQAT_UNKNOWN
- MQAT_USER_FIRST
- MQAT_USER_LAST

Sie können auch eine anwendungsspezifische Zeichenfolge Ihrer Wahl verwenden. Weitere Informationen hierzu finden Sie im Feld „PutApplType (MQLONG)" auf Seite 433 des Nachrichtendeskriptors (MQMD).

put date
Das Datum, an dem eine Nachricht eingereiht wurde. Der Anfangswert ist eine Nullzeichenfolge.

put time
Die Uhrzeit, zu der eine Nachricht eingereiht wurde. Der Anfangswert ist eine Nullzeichenfolge.

reply-to queue manager name
Der Name des Warteschlangenmanagers, an den alle Antworten gesendet werden sollen. Der Anfangswert ist eine Nullzeichenfolge.

reply-to queue name
Der Name der Warteschlange, an die alle Antworten gesendet werden sollen. Der Anfangswert ist eine Nullzeichenfolge.

report
Die einer Nachricht zugeordneten Rückmeldeinformationen. Der Anfangswert ist MQRO_NONE. Folgende zusätzliche Werte sind möglich:

- 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

wobei * Werte angibt, die unter WebSphere MQ for z/OS nicht unterstützt werden.

sequence number
Angaben zur Reihenfolge für die Ermittlung einer Nachricht innerhalb einer Gruppe. Der Anfangswert ist eins.

total message length
Die Anzahl der beim aktuellsten Auslesen einer Nachricht verfügbaren Bytes. Diese Anzahl ist höher als die Nachrichtenlänge (**message length**) von "ImqCache", wenn die letzte Nachricht abgeschnitten

wurde oder wenn die letzte Nachricht nicht ausgelesen wurde, weil sie sonst abgeschnitten worden wäre. Dieses Attribut ist schreibgeschützt. Der Anfangswert ist null.

Dieses Attribut kann in allen Situationen, in denen abgeschnittene Nachrichten eine Rolle spielen, nützlich sein.

user id
Die einer Nachricht zugeordnete Benutzeridentität. Der Anfangswert ist eine Nullzeichenfolge.

Konstruktoren

ImqMessage( );
Der Standardkonstruktor.

ImqMessage(const ImqMessage & *Nachricht*);
Der Kopierkonstruktor. Weitere Informationen finden Sie in der Beschreibung der Methode **operator =**.

Objektmethoden (öffentlich)

void operator = (const ImqMessage & *Nachricht*);
Kopiert die MQMD- und Nachrichtendaten aus *msg*. Wenn der Benutzer einen Puffer für dieses Objekt bereitgestellt hat, wird das kopierte Datenvolumen auf die verfügbare Puffergröße begrenzt. Andernfalls stellt das System sicher, dass ein Puffer von ausreichender Größe für die kopierten Daten verfügbar ist

ImqString applicationIdData( ) const ;
Gibt eine Kopie der ID-Daten der Anwendung (**application id data**) zurück.

void setApplicationIdData(const char * *data* = 0);
Legt die ID-Daten der Anwendung (**application id data**) fest.

ImqString applicationOriginData( ) const ;
Gibt eine Kopie der Ursprungsdaten der Anwendung (**application origin data**) zurück.

void setApplicationOriginData(const char * *data* = 0);
Legt die Ursprungsdaten der Anwendung (**application origin data**) fest.

MQLONG backoutCount( ) const ;
Gibt den Rücksetzungszähler (**backout count**) zurück.

MQLONG characterSet( ) const ;
Gibt den Zeichensatz (**character set**) zurück.

void setCharacterSet(const MQLONG *ccsid* = MQCCSI_Q_MGR);
Legt den Zeichensatz (**character set**) fest.

MQLONG encoding( ) const ;
Gibt die Codierung (**encoding**) zurück.

void setEncoding(const MQLONG *encoding* = MQENC_NATIVE);
Legt die Codierung (**encoding**) fest.

MQLONG expiry( ) const ;
Gibt die Ablaufzeit (**expiry**) zurück.

void setExpiry(const MQLONG *expiry*);
Legt die Ablaufzeit (**expiry**) fest.

ImqString format( ) const ;
Gibt eine Kopie des Formats (**format**) zurück, einschließlich abschließender Leerzeichen.

ImqBoolean formatIs(const char * *format-to-test*) const ;
Gibt TRUE zurück, wenn das **format** mit dem zu testenden Format (*format-to-test*) übereinstimmt.

void setFormat(const char * *name* = 0);
Legt das **Format**fest, das auf acht Zeichen mit abschließenden Leerzeichen aufgefüllt wird.

MQLONG messageFlags( ) const ;
Gibt die Nachrichtenflags (**message flags**) zurück.

void setMessageFlags(const MQLONG *flags*);
Legt die Nachrichtenflags (**message flags**) fest.

MQLONG messageType( ) const ;
Gibt den Nachrichtentyp (**message type**) zurück.

void setMessageType(const MQLONG *type*);
Legt den Nachrichtentyp (**message type**) fest.

MQLONG offset( ) const ;
Gibt die relative Adresse (**offset**) zurück.

void setOffset(const MQLONG *offset*);
Legt die relative Adresse (**offset**) fest.

MQLONG originalLength( ) const ;
Gibt die ursprüngliche Länge (**original length**) zurück.

void setOriginalLength(const MQLONG *length*);
Legt die ursprüngliche Länge (**original length**) fest.

MQLONG persistence( ) const ;
Gibt die Persistenz (**persistence**) zurück.

void setPersistence(const MQLONG *persistence*);
Legt die Persistenz (**persistence**) fest.

MQLONG priority( ) const ;
Gibt die Priorität (**priority**) zurück.

void setPriority(const MQLONG *priority*);
Legt die Priorität (**priority**) fest.

ImqString putApplicationName( ) const ;
Gibt eine Kopie des Namens der PUT-Anwendung (**put application name**) zurück.

void setPutApplicationName(const char * *name* = 0);
Legt den Namen der PUT-Anwendung (**put application name**) fest.

MQLONG putApplicationType( ) const ;
Gibt den Typ der PUT-Anwendung (**put application type**) zurück.

void setPutApplicationType(const MQLONG *type* = MQAT_NO_CONTEXT);
Legt den Typ der PUT-Anwendung (**put application type**) fest.

ImqString putDate( ) const ;
Gibt eine Kopie des PUT-Datums (**put date**) zurück.

void setPutDate(const char * *date* = 0);
Legt das PUT-Datum (**put date**) fest.

ImqString putTime( ) const ;
Gibt eine Kopie der PUT-Zeit (**put time**) zurück.

void setPutTime(const char * *time* = 0);
Legt die PUT-Zeit (**put time**) fest.

ImqBoolean readItem(ImqItem & *Element*);
Liest aus dem Nachrichtenpuffer in das Objekt *item* und verwendet dabei die "ImqItem"-Methode **pasteIn**. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString replyToQueueManagerName( ) const ;
Gibt eine Kopie des Namens des Managers der Empfangswarteschlange für Antworten (**reply-to queue name**) zurück.

void setReplyToQueueManagerName(const char * *name* = 0);
Legt den Namen des Managers der Empfangswarteschlange für Antworten (**reply-to queue manager name**) fest.

ImqString replyToQueueName( ) const ;
Gibt eine Kopie des Namens der Empfangswarteschlange für Antworten (**reply-to queue name**) zurück.

void setReplyToQueueName(const char * *name* = 0);
Legt den Namen der Empfangswarteschlange für Antworten (**reply-to queue name**) fest.

MQLONG report( ) const ;
Gibt den Bericht (**report**) zurück.

void setReport(const MQLONG *report*);
Legt den Bericht (**report**) fest.

MQLONG sequenceNumber( ) const ;
Gibt die Folgenummer (**sequence number**) zurück.

void setSequenceNumber(const MQLONG *number*);
Legt die Folgenummer (**sequence number**) fest.

size_t totalMessageLength( ) const ;
Gibt die Gesamtlänge der Nachricht (**total message length**) zurück.

ImqString userId( ) const ;
Gibt eine Kopie der Benutzer-ID (**user id**) zurück.

void setUserId(const char * *id* = 0);
Legt die Benutzer-ID (**user id**) fest.

ImqBoolean writeItem(ImqItem & *Element*);
Schreibt aus dem Objekt *item* in den Nachrichtenpuffer und verwendet dabei die "ImqItem"-Methode **copyOut**. Der Schreibvorgang kann durch Einfügen, Ersetzen oder Hinzufügen erfolgen; dies hängt von der Klasse des Objekts *item* ab. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Objektmethoden (geschützt)

static void setVersionSupported(const MQLONG);
Legt den **MQMD-Version** fest. Nimmt standardmäßig den Wert **MQMD_VERSION_2** an.

Objektdaten (geschützt)

MQMD1 *omqmd*
(Nur WebSphere MQ für z/OS .) Die MQMD-Datenstruktur.

MQMD2 *omqmd*
(Andere Plattformen als z/OS.) Die MQMD-Datenstruktur.

C++-Klasse "ImqMessageTracker"

Diese Klasse bindet diejenigen Attribute eines "ImqMessage"- oder "ImqQueue"-Objekts ein, die einem der beiden Objekte zugeordnet werden können.

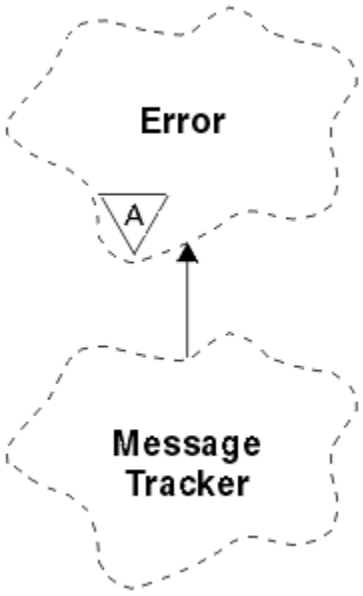


Abbildung 59. ImqMessageTracker, Klasse

Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqMessageTracker” auf Seite 1350 aufgelisteten MQI-Aufrufe.

Objektattribute

accounting token
Ein binärer Wert (MQBYTE32) mit der Länge MQ_ACCOUNTING_TOKEN_LENGTH. Der Anfangswert ist MQACT_NONE.

correlation id
Ein binärer Wert (MQBYTE24) mit der Länge MQ_CORREL_ID_LENGTH, den Sie zum Korrelieren von Nachrichten zuordnen. Der Anfangswert ist MQCI_NONE. Der zusätzliche Wert MQCI_NEW_SESSION ist möglich.

feedback
Rückmeldeinformationen, die zusammen mit einer Nachricht gesendet werden. Der Anfangswert ist MQFB_NONE. Folgende zusätzliche Werte sind möglich:

- MQFB_SYSTEM_FIRST
- 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_NEGATIVE
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IMS
- MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_BRIDGE_FAILURE
- MQFB_CICS_CCSID_ERROR
- MQFB_CICS_CIH_ERROR
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_CORREL_ID_ERROR
- MQFB_CICS_DLQ_ERROR
- MQFB_CICS_ENCODING_ERROR
- MQFB_CICS_INTERNAL_ERROR
- MQFB_CICS_NOT_AUTHORIZED
- MQFB_CICS_UOW_BACKED_OUT
- MQFB_CICS_UOW_ERROR

Sie können auch eine anwendungsspezifische Zeichenfolge Ihrer Wahl verwenden. Weitere Informationen hierzu finden Sie im Feld „Feedback (MQLONG)" auf Seite 411 des Nachrichtendeskriptors (MQMD).

group id
Ein innerhalb einer Warteschlange eindeutiger binärer Wert (MQBYTE24) mit der Länge MQ_GROUP_ID_LENGTH. Der Anfangswert ist MQGI_NONE.

message id
Ein innerhalb einer Warteschlange eindeutiger binärer Wert (MQBYTE24) mit der Länge MQ_MSG_ID_LENGTH. Der Anfangswert ist MQMI_NONE.

Konstruktoren

ImqMessageTracker( );
Der Standardkonstruktor.

ImqMessageTracker (const ImqMessageTracker & *Tracker*)
Der Kopierkonstruktor. Weitere Informationen finden Sie in der Beschreibung der Methode **operator =**.

Objektmethoden (öffentlich)

void operator = (const ImqMessageTracker & *Tracker*);
Kopiert Instanzdaten aus *tracker* und ersetzt dabei die bereits bereits vorhandenen Instanzdaten.

ImqBinary accountingToken( ) const ;
Gibt eine Kopie des Abrechnungstokens (**accounting token**) zurück.

ImqBoolean setAccountingToken(const ImqBinary & *Token*);
Legt das Abrechnungstoken (**accounting token**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_ACCOUNTING_TOKEN_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setAccountingToken(const MQBYTE32 *token* = 0);
Legt das Abrechnungstoken (**accounting token**) fest. Der Wert für *token* kann null sein, was der Angabe von MQACT_NONE entspricht. Wenn der Wert für *token* ungleich null ist, muss er MQ_ACCOUNTING_TOKEN_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQACT_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQACT_NONE.

ImqBinary correlationId( ) const ;
Gibt eine Kopie der Korrelations-ID (**correlation id**) zurück.

ImqBoolean setCorrelationId(const ImqBinary & *Token*);
Legt die Korrelations-ID (**correlation id**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_CORREL_ID_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setCorrelationId(const MQBYTE24 *id* = 0);
Legt die Korrelations-ID (**correlation id**) fest. Der Wert für *id* kann null sein, was der Angabe von MQCI_NONE entspricht. Wenn der Wert für *id* ungleich null ist, muss er MQ_CORREL_ID_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQCI_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQCI_NONE.

MQLONG feedback( ) const ;
Gibt das **Feedback** zurück.

void setFeedback(const MQLONG *feedback*);
Legt das **Feedback** fest.

ImqBinary groupId( ) const ;
Gibt eine Kopie der Gruppen-ID (**group id**) zurück.

ImqBoolean setGroupId(const ImqBinary & *Token*);
Legt die Gruppen-ID (**group id**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_GROUP_ID_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setGroupId(const MQBYTE24 *id* = 0);
Legt die Gruppen-ID (**group id**) fest. Der Wert für *id* kann null sein, was der Angabe von MQGI_NONE entspricht. Wenn der Wert für *id* ungleich null ist, muss er MQ_GROUP_ID_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQGI_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQGI_NONE.

ImqBinary messageId( ) const ;
Gibt eine Kopie der Nachrichten-ID (**message id**) zurück.

ImqBoolean setMessageId(const ImqBinary & *Token*);
Legt die Nachrichten-ID (**message id**) fest. Die Datenlänge (**data length**) von *token* muss entweder null oder MQ_MSG_ID_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setMessageId(const MQBYTE24 *id* = 0);
Legt die Nachrichten-ID (**message id**) fest. Der Wert für *id* kann null sein, was der Angabe von MQMI_NONE entspricht. Wenn der Wert für *id* ungleich null ist, muss er MQ_MSG_ID_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQMI_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQMI_NONE.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR

C++-Klasse "ImqNamelist"

Diese Klasse bindet eine Namensliste ein.

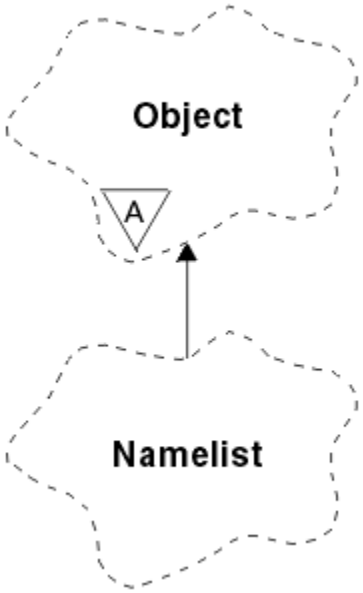


Abbildung 60. ImqNamelist, Klasse

Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqNamelist” auf Seite 1351 aufgelisteten MQI-Aufrufe.

Objektattribute

name count
Die Anzahl der Objektnamen in den Namenslistennamen (**namelist names**). Dieses Attribut ist schreibgeschützt.

namelist names
Objektnamen, deren Anzahl vom Namenszähler (**name count**) angegeben wird. Dieses Attribut ist schreibgeschützt.

Konstruktoren

ImqNamelist( );
Der Standardkonstruktor.

ImqNamelist(const ImqNamelist & ***Liste*** **);**
Der Kopierkonstruktor. Der Öffnungsstatus (**open status**) von "ImqObject" ist auf FALSE gesetzt.

ImqNamelist(const char * ***name*****);**
Setzt den "ImqObject"-Namen auf **name**.

Objektmethoden (öffentlich)

void operator = (const ImqNamelist & ***Liste*** **);**
Kopiert Instanzdaten aus *list* und ersetzt dabei die bereits vorhandenen Instanzdaten. Der Öffnungsstatus (**open status**) von "ImqObject" ist auf FALSE gesetzt.

ImqBoolean nameCount(MQLONG & *anzahl*);
Stellt eine Kopie des Namenszählers (**name count**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG nameCount ( );
Gibt den Namenszähler (**name count**) ohne Angabe möglicher Fehler zurück.

ImqBoolean namelistName (const MQLONG *index*, ImqString & *name*);
Stellt eine Kopie eines der Namenslistennamen (**namelist names**) über einen auf null basierenden Index bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString namelistName (const MQLONG *index*);
Gibt einen der Namenslistennamen (**namelist names**) über einen auf null basierenden Index ohne Angabe möglicher Fehler zurück.

Ursachencodes

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

C++-Klasse "ImqObject"

Diese Klasse ist abstrakt. Wenn ein Objekt dieser Klasse gelöscht wird, wird es automatisch geschlossen und seine "ImqQueueManager"-Verbindung wird getrennt.

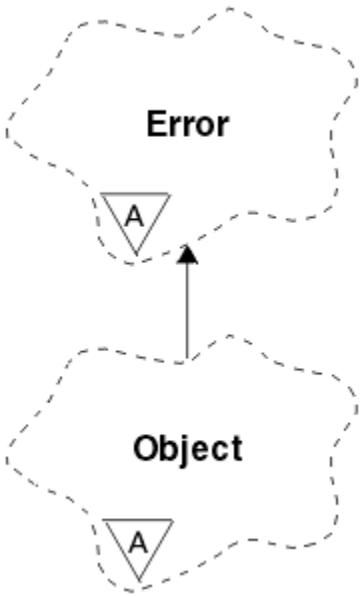


Abbildung 61. ImqObject, Klasse

Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqObject” auf Seite 1351 aufgelisteten MQI-Aufrufe.

•

Klassenattribute

behavior
Steuert das Verhalten von implizitem Öffnen.

IMQ_IMPL_OPEN (8L)
Implizites Öffnen ist zulässig. Dies ist die Standardeinstellung.

Objektattribute

alteration date
Das Änderungsdatum. Dieses Attribut ist schreibgeschützt.

alteration time
Die Änderungszeit. Dieses Attribut ist schreibgeschützt.

alternate user id
Die alternative Benutzer-ID mit bis zu MQ_USER_ID_LENGTH Zeichen. Der Anfangswert ist eine Nullzeichenfolge.

alternate security id
Die alternative Sicherheits-ID. Ein binärer Wert (MQBYTE40) mit der Länge MQ_SECURITY_ID_LENGTH. Der Anfangswert ist MQSID_NONE.

close options
Optionen, die beim Schließen eines Objekts Anwendung finden. Der Anfangswert ist MQCO_NONE. Dieses Attribut wird bei Operationen zum impliziten erneuten Öffnen ignoriert, bei denen stets ein MQCO_NONE-Wert verwendet wird.

connection reference
Ein Verweis auf ein ImqQueueManager-Objekt, das die erforderliche Verbindung zu einem (lokalen) Warteschlangenmanager bereitstellt. Für ein ImqQueueManager-Objekt ist dies das Objekt selbst. Der Anfangswert ist null.

Anmerkung: Verwechseln Sie diesen Verweis nicht mit dem Warteschlangenmanagernamen (**queue manager name**), der einen (möglicherweise fernen) Warteschlangenmanager für eine benannte Warteschlange angibt.

description
Der beschreibende Name (bis zu 64 Zeichen lang) des Warteschlangenmanagers, der Warteschlange, der Namensliste oder des Prozesses. Dieses Attribut ist schreibgeschützt.

name
Der Name (bis zu 48 Zeichen lang) des Warteschlangenmanagers, der Warteschlange, der Namensliste oder des Prozesses. Der Anfangswert ist eine Nullzeichenfolge. Der Name einer Modellwarteschlange wird nach einem Öffnungsvorgang (**open**) in den Namen der sich ergebenden dynamischen Warteschlange geändert.

Anmerkung: Ein ImqQueueManager-Warteschlangenmanager kann einen Nullnamen aufweisen, der den Standardwarteschlangenmanager darstellt. Nach einem erfolgreichen Öffnen (**open**) ändert sich der Name in den des aktuellen Warteschlangenmanagers. Eine Verteilerliste "ImqDistributionList" ist dynamisch und muss einen Nullnamen aufweisen.

next managed object
Dies ist das nächste Objekt dieser Klasse in nicht festgelegter Reihenfolge, das denselben Verbindungsverweis (**connection reference**) wie dieses Objekt aufweist. Der Anfangswert ist null.

open options
Optionen, die beim Öffnen eines Objekts Anwendung finden. Der Anfangswert ist MQOO_INQUIRE. Es gibt zwei Möglichkeiten zum Setzen geeigneter Werte:

1. Setzen Sie nicht die Optionen zum Öffnen (**open options**) und verwenden Sie nicht die Methode **open**. WebSphere MQ passt die Optionen zum Öffnen (**open options**) automatisch an und öffnet und schließt Objekte automatisch nach Bedarf. Dies kann unnötige Operationen zum erneuten

Öffnen zur Folge haben, weil WebSphere MQ die Methode **openFor** verwendet und diese die Optionen zum Öffnen (**open options**) nur schrittweise hinzufügt.

2. Setzen Sie die Optionen zum Öffnen (**open options**), bevor Sie Methoden verwenden, die einen MQI-Aufruf auslösen (siehe „Querverweise zwischen C++ und MQI" auf Seite 1345). Dies stellt sicher, dass keine unnötigen Operationen zum erneuten Öffnen auftreten. Setzen Sie Optionen zum Öffnen explizit, wenn davon auszugehen ist, dass potenzielle Probleme mit dem erneuten Öffnen auftreten werden (siehe Erneutes Öffnen).

   Wenn Sie die Methode **open** verwenden, ***müssen*** Sie sicherstellen, dass zuvor geeignete Optionen zum Öffnen (**open options**) festgelegt werden. Die Verwendung der Methode **open** ist jedoch nicht obligatorisch; WebSphere MQ weist dann immer noch dasselbe Verhalten auf wie in Fall 1, doch unter den beschriebenen Umständen ist das Verhalten effizient.

Null ist kein gültiger Wert; setzen Sie den entsprechenden Wert, bevor Sie versuchen, das Objekt zu öffnen. Dies erfolgt entweder durch Verwendung von **setOpenOptions**(*lOpenOptions*) und anschließend **open**( ) oder durch Verwendung von **openFor**(*lRequiredOpenOption*).

Anmerkung:

1. MQOO_OUTPUT wird für MQOO_INQUIRE während der Methode **open** für eine Verteilerliste ersetzt, da MQOO_OUTPUT zu diesem Zeitpunkt die einzige gültige Option zum Öffnen (**open option**) ist. Dennoch hat es sich bewährt, MQOO_OUTPUT stets explizit in Anwendungsprogrammen zu setzen, die die Methode **open** verwenden.
2. Geben Sie MQOO_RESOLVE_NAMES an, wenn Sie die Attribute **resolved queue manager name** (aufgelöster Warteschlangenmanagername) und **resolved queue name** (aufgelöster Warteschlangenname) der Klasse verwenden wollen.

open status
Gibt an, ob das Objekt offen (TRUE) oder geschlossen (FALSE) ist. Der Anfangswert ist FALSE. Dieses Attribut ist schreibgeschützt.

previous managed object
Das vorherige Objekt dieser Klasse in nicht festgelegter Reihenfolge, das denselben Verbindungsverweis (**connection reference**) wie dieses Objekt aufweist. Der Anfangswert ist null.

queue manager identifier
Die Warteschlangenmanager-ID. Dieses Attribut ist schreibgeschützt.

Konstruktoren

ImqObject( );
Der Standardkonstruktor.

ImqObject(const ImqObject & *Objekt*);
Der Kopierkonstruktor. Der Öffnungsstatus (**open status**) ist dann auf FALSE gesetzt.

Klassenmethoden (öffentlich)

static MQLONG behavior( );
Gibt das Verhalten (**behavior**) zurück.

void setBehavior(const MQLONG *behavior* = 0);
Legt das Verhalten (**behavior**) fest.

Objektmethoden (öffentlich)

void operator = (const ImqObject & *objekt*);
Führt bei Bedarf einen Schließvorgang aus und kopiert die Instanzdaten aus *object*. Der Öffnungsstatus (**open status**) ist dann auf FALSE gesetzt.

ImqBoolean alterationDate(ImqString & *Datum*);
Stellt eine Kopie des Änderungsdatums (**alteration date**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString alterationDate( );
Gibt das Änderungsdatum (**alteration date**) ohne Angabe möglicher Fehler zurück.

ImqBoolean alterationTime(ImqString & ***Zeit*** **);**
Stellt eine Kopie der Änderungszeit (**alteration time**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString alterationTime( );
Gibt die Änderungszeit (**alteration time**) ohne Angabe möglicher Fehler zurück.

ImqString alternateUserId( ) const ;
Gibt eine Kopie der alternativen Benutzer-ID (**alternate user id**) zurück.

ImqBoolean setAlternateUserId(const char * ***id*** **);**
Legt die alternative Benutzer-ID (**alternate user id**) fest. Die alternative Benutzer-ID (**alternate user id**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) auf FALSE gesetzt ist. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBinary alternateSecurityId( ) const ;
Gibt eine Kopie der alternativen Sicherheits-ID (**alternate security id**) zurück.

ImqBoolean setAlternateSecurityId(const ImqBinary & ***Token*** **);**
Legt die alternative Sicherheits-ID (**alternate security id**) fest. Die alternative Sicherheits-ID (**alternate security id**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) auf FALSE gesetzt ist. Die Datenlänge von *token* muss entweder null oder MQ_SECURITY_ID_LENGTH sein. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean setAlternateSecurityId(const MQBYTE* ***token*** **= 0);**
Legt die alternative Sicherheits-ID (**alternate security id**) fest. Der Wert für *token* kann null sein, was der Angabe von MQSID_NONE entspricht. Wenn der Wert für *token* ungleich null ist, muss er MQ_SECURITY_ID_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQSID_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQSID_NONE.

Die alternative Sicherheits-ID (**alternate security id**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) auf TRUE gesetzt ist. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean setAlternateSecurityId(const unsigned char * ***id*** **= 0);**
Legt die alternative Sicherheits-ID (**alternate security id**) fest.

ImqBoolean close( );
Setzt den Öffnungsstatus (**open status**) auf FALSE. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG closeOptions( ) const ;
Gibt die Optionen zum Schließen (**close options**) zurück.

void setCloseOptions(const MQLONG ***options*** **);**
Legt die Optionen zum Schließen (**close options**) fest.

ImqQueueManager * connectionReference( ) const ;
Gibt den Verbindungsverweis (**connection reference**) zurück.

void setConnectionReference(ImqQueueManager & ***manager*** **)**
Legt den Verbindungsverweis (**connection reference**) fest.

void setConnectionReference(ImqQueueManager * ***manager*** **= 0);**
Legt den Verbindungsverweis (**connection reference**) fest.

virtuelle ImqBoolean Beschreibung(ImqString & ***Beschreibung*** **) = 0;**
Stellt eine Kopie der Beschreibung (**description**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString description( );
Gibt eine Kopie der Beschreibung (**description**) ohne Angabe möglicher Fehler zurück.

virtuell ImqBoolean name(ImqString & ***name*** **);**
Stellt eine Kopie des Namens (**name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString name( );
Gibt eine Kopie des Namens (**name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setName(const char * *name* = 0);
Legt den Namen (**name**) fest. Der Name (**name**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) auf FALSE gesetzt ist, und im Falle eines Warteschlangenmanagers ImqQueueManager, wenn der Verbindungsstatus (**connection status**) auf FALSE gesetzt ist. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqObject * nextManagedObject( ) const ;
Gibt das nächste verwaltete Objekt (**next managed object**) zurück.

ImqBoolean open( );
Ändert den Öffnungsstatus (**open status**) in TRUE, indem das Objekt bei Bedarf geöffnet wird, wobei neben anderen Attributen die Optionen zum Öffnen (**open options**) und der Name (**name**) verwendet werden. Diese Methode verwendet die Information zum Verbindungsverweis (**connection reference**) und die ImqQueueManager-Methode **connect**, um sicherzustellen, dass der Verbindungsstatus (**connection status**) von ImqQueueManager auf TRUE gesetzt ist. Gibt den Öffnungsstatus (**open status**) zurück.

ImqBoolean openFor(const MQLONG *required-options* = 0);
Versucht, sicherzustellen, dass das Objekt mit Optionen zum Öffnen (**open options**) bzw. mit Optionen zum Öffnen (**open options**), die das vom Parameterwert *required-options* implizierte Verhalten garantieren, geöffnet ist.

Wenn der Wert für *required-options* null ist, ist eine Eingabe erforderlich und jede beliebige Eingabeoption ist dabei ausreichend. Weisen also die Optionen zum Öffnen (**open options**) bereits eine der folgenden Optionen auf:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

dann sind die Optionen zum Öffnen (**open options**) bereits zufriedenstellend und werden nicht geändert; wenn die Optionen zum Öffnen (**open options**) dagegen keine dieser Optionen enthalten, wird MQOO_INPUT_AS_Q_DEF in den Optionen zum Öffnen (**open options**) gesetzt.

Wenn der Wert für *required-options* ungleich null ist, werden die erforderlichen Optionen den Optionen zum Öffnen (**open options**) hinzugefügt; wenn der Wert für *required-options* eine dieser Optionen ist, werden die anderen zurückgesetzt.

Wenn irgendwelche Optionen zum Öffnen (**open options**) geändert werden und das Objekt bereits geöffnet ist, wird das Objekt vorübergehend geschlossen und dann erneut geöffnet, um die Optionen zum Öffnen (**open options**) anzupassen.

Bei erfolgreicher Ausführung wird TRUE zurückgegeben. Erfolgreich bedeutet, dass das Objekt mit den entsprechenden Optionen geöffnet ist.

MQLONG openOptions( ) const ;
Gibt die Optionen zum Öffnen (**open options**) zurück.

ImqBoolean setOpenOptions(const MQLONG *options*);
Legt die Optionen zum Öffnen (**open options**) fest. Die Optionen zum Öffnen (**open options**) können nur festgelegt werden, wenn der Öffnungsstatus (**open status**) auf FALSE gesetzt ist. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean openStatus( ) const ;
Gibt den Öffnungsstatus (**open status**) zurück.

ImqObject * previousManagedObject( ) const ;
Gibt das vorherige verwaltete Objekt (**previous managed object**) zurück.

ImqBoolean queueManagerIdentifier (ImqString & *ID*);
Stellt eine Kopie der Warteschlangenmanager-ID (**queue manager identifier**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString queueManagerIdentifier( );
Gibt die Warteschlangenmanager-ID (**queue manager identifier**) ohne Angabe möglicher Fehler zurück.

Objektmethoden (geschützt)

virtual ImqBoolean closeTemporarily( );
Schließt ein Objekt sicher, bevor es erneut geöffnet wird. Bei erfolgreicher Ausführung wird TRUE zurückgegeben. Diese Methode setzt voraus, dass der Öffnungsstatus (**open status**) auf TRUE gesetzt ist.

MQHCONN connectionHandle( ) const ;
Gibt die Verbindungskennung MQHCONN zurück, die dem Verbindungsverweis (**connection reference**) zugeordnet ist. Dieser Wert ist null, wenn kein Verbindungsverweis (**connection reference**) vorhanden ist oder wenn keine Verbindung zum Manager besteht.

ImqBoolean inquire(const MQLONG *int-attr*, MQLONG & *wert*);
Gibt einen ganzzahligen Wert zurück, dessen Index ein MQIA_*-Wert ist. Bei einem Fehler wird der Wert auf MQIAV_UNDEFINED zurückgesetzt.

ImqBoolean inquire(const MQLONG *char-attr*, char * & *buffer*, const size_t *länge*);
Gibt eine Zeichenfolge zurück, deren Index ein MQCA_*-Wert ist.

Anmerkung: Beide Methoden geben nur einen einzelnen Attributwert zurück. Ist eine *Momentaufnahme* von mehreren Werten erforderlich, bei der die Werte für einen Moment miteinander konsistent sind, stellt WebSphere MQ C++ diese Funktion nicht bereit und Sie müssen den MQINQ-Aufruf mit entsprechenden Parametern nutzen.

virtual void openInformationDisperse( );
Verbreitet unmittelbar nach einem MQOPEN-Aufruf Informationen aus dem Variablenabschnitt der MQOD-Datenstruktur.

virtual ImqBoolean openInformationPrepare( );
Bereitet unmittelbar vor einem MQOPEN-Aufruf Informationen für den Variablenabschnitt der MQOD-Datenstruktur vor und gibt nach erfolgreicher Ausführung TRUE zurück.

ImqBoolean set(const MQLONG *int-attr*, const MQLONG *value*);
Legt ein WebSphere MQ-Ganzzahlenattribut fest.

ImqBoolean set(const MQLONG *char-attr*, const char * *buffer*, const size_t *required-length*);
Legt ein WebSphere MQ-Zeichenattribut fest.

void setNextManagedObject(const ImqObject * *object* = 0);
Legt das nächste verwaltete Objekt (**next managed object**) fest.

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass die Liste der verwalteten Objekte dadurch nicht unterbrochen wird.

void setPreviousManagedObject(const ImqObject * *object* = 0);
Legt das vorherige verwaltete Objekt (**previous managed object**) fest.

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass die Liste der verwalteten Objekte dadurch nicht unterbrochen wird.

Objektdaten (geschützt)

MQHOBJ *ohobj*
Die WebSphere MQ-Objektkennung (nur gültig, wenn der Öffnungsstatus (**open status**) auf TRUE gesetzt ist).

MQOD *omqod*
Die eingebettete MQOD-Datenstruktur. Der für diese Datenstruktur zugeordnete Speicher entspricht dem, der für die Version 2 einer MQOD erforderlich ist. Überprüfen Sie die Versionsnummer *(omqod.Version)* und greifen Sie auf die anderen Felder wie folgt zu:

MQOD_VERSION_1
Auf alle anderen Felder in *omqod* kann zugegriffen werden.

MQOD_VERSION_2
Auf alle anderen Felder in *omqod* kann zugegriffen werden.

MQOD_VERSION_3
omqod.pmqod ist ein Verweis auf eine dynamisch zugeordnete, umfangreichere MQOD. Auf andere Felder in *omqod* kann nicht zugegriffen werden. Auf alle von *omqod.pmqod* adressierten Felder kann zugegriffen werden.

Anmerkung: *omqod.pmqod.Version* kann niedriger sein als *omqod.Version*, was anzeigt, dass der WebSphere MQ-MQI-Client mehr Funktionen aufweist als der WebSphere MQ-Server.

Ursachencodes

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (Ursachencodes von MQCLOSE)
- (Ursachencodes von MQCONN)
- (Ursachencodes von MQINQ)
- (Ursachencodes von MQOPEN)
- (Ursachencodes von MQSET)

C++-Klasse "ImqProcess"

Diese Klasse bindet einen Anwendungsprozess (ein WebSphere MQ-Objekt des Typs MQOT_PROCESS) ein, der von einem Auslösemonitor ausgelöst werden kann.

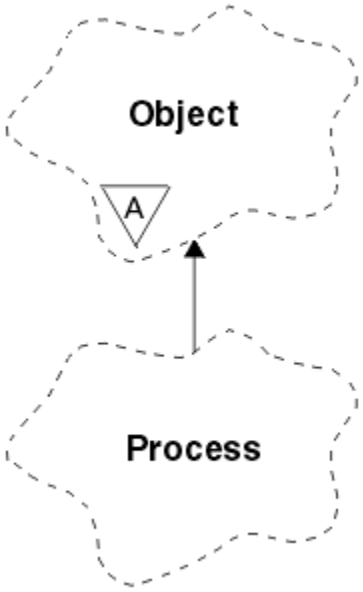


Abbildung 62. ImqProcess, Klasse

- „Objektattribute” auf Seite 1412
- „Konstruktoren” auf Seite 1412

- „Objektmethoden (öffentlich)” auf Seite 1412

Objektattribute

application id
Die ID des Anwendungsprozesses. Dieses Attribut ist schreibgeschützt.

application type
Der Typ des Anwendungsprozesses. Dieses Attribut ist schreibgeschützt.

environment data
Die Umgebungsinformationen für diesen Prozess. Dieses Attribut ist schreibgeschützt.

user data
Benutzerdaten für den Prozess. Dieses Attribut ist schreibgeschützt.

Konstruktoren

ImqProcess( );
Der Standardkonstruktor.

ImqProcess(const ImqProcess & ***Prozess*** **);**
Der Kopierkonstruktor. Der Öffnungsstatus (**open status**) von "ImqObject" ist auf FALSE gesetzt.

ImqProcess(const char * ***name*** **);**
Legt den Namen (**name**) von "ImqObject" fest.

Objektmethoden (öffentlich)

void operator = (const ImqProcess & ***Prozess*** **);**
Führt bei Bedarf einen Schließvorgang aus und kopiert anschließend Instanzdaten aus *process*. Der Öffnungsstatus (**open status**) von "ImqObject" ist dann auf FALSE gesetzt.

ImqBoolean applicationId(ImqString & ***ID*** **);**
Stellt eine Kopie der Anwendungs-ID (**application id**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString applicationId( );
Gibt die Anwendungs-ID (**application id**) ohne Angabe möglicher Fehler zurück.

ImqBoolean applicationType(MQLONG & ***typ*** **);**
Stellt eine Kopie des Anwendungstyps (**application type**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG applicationType( );
Gibt den Anwendungstyp (**application type**) ohne Angabe möglicher Fehler zurück.

ImqBoolean environmentData(ImqString & ***data*** **);**
Stellt eine Kopie der Umgebungsdaten (**environment data**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString environmentData( );
Gibt die Umgebungsdaten (**environment data**) ohne Angabe möglicher Fehler zurück.

ImqBoolean userData(ImqString & ***Daten*** **);**
Stellt eine Kopie der Benutzerdaten (**user data**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString userData( );
Gibt die Benutzerdaten (**user data**) ohne Angabe möglicher Fehler zurück.

C++-Klasse "ImqPutMessageOptions"

Diese Klasse bindet die MQPMO-Datenstruktur ein.

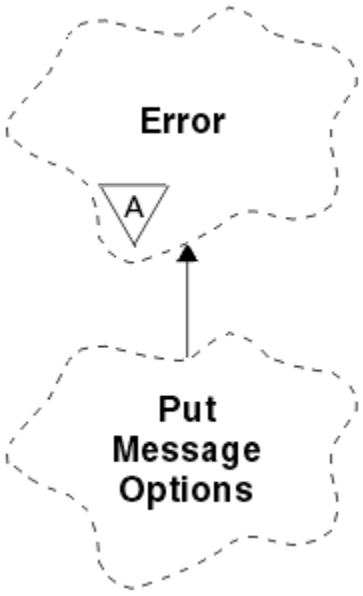


Abbildung 63. ImqPutMessageOptions, Klasse

Objektattribute

context reference
Eine "ImqQueue"-Warteschlange, die einen Kontext für Nachrichten bereitstellt. Zu Beginn ist kein Verweis vorhanden.

options
Die Optionen zum Einreihen von Nachrichten. Der Anfangswert ist MQPMO_NONE. Folgende zusätzliche Werte sind möglich:

- 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

record fields
Die Flags, die beim Einreihen einer Nachricht die Aufnahme von Nachrichteneinreihungssätzen steuern. Der Anfangswert ist MQPMRF_NONE. Folgende zusätzliche Werte sind möglich:

- MQPMRF_MSG_ID
- MQPMRF_CORREL_ID
- MQPMRF_GROUP_ID
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN

"ImqMessageTracker"-Attribute werden für jedes angegebene Feld aus dem Objekt übernommen. "ImqMessageTracker"-Attribute werden für jedes *nicht* angegebene Feld aus dem "ImqMessage"-Objekt übernommen.

resolved queue manager name
Name eines Zielwarteschlangenmanagers, der während einer Einreihung festgelegt wird. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

resolved queue name
Name einer Zielwarteschlange, der während einer Einreihung festgelegt wird. Der Anfangswert ist null. Dieses Attribut ist schreibgeschützt.

syncpoint participation
Auf TRUE gesetzt, wenn Nachrichten unter Synchronisationspunktsteuerung eingereiht werden.

Konstruktoren

ImqPutMessageOptions( );
Der Standardkonstruktor.

ImqPutMessageOptions(const ImqPutMessageOptions & *pmo*);
Der Kopierkonstruktor.

Objektmethoden (öffentlich)

void operator = (const ImqPutMessageOptions & *pmo*);
Kopiert Instanzdaten aus *pmo* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqQueue * contextReference( ) const ;
Gibt den Kontextverweis (**context reference**) zurück.

void setContextReference(const ImqQueue & *Warteschlange*);
Legt den Kontextverweis (**context reference**) fest.

void setContextReference(const ImqQueue * *queue* = 0);
Legt den Kontextverweis (**context reference**) fest.

MQLONG options( ) const ;
Gibt die Optionen (**options**) zurück.

void setOptions(const MQLONG *options*);
Legt die Optionen (**options**) fest, einschließlich des Werts für die Synchronisationspunktbeteiligung (**syncpoint participation**).

MQLONG recordFields( ) const ;
Gibt die Datensatzfelder (**record fields**) zurück.

void setRecordFields(const MQLONG *fields*);
Legt die Datensatzfelder (**record fields**) fest.

ImqString resolvedQueueManagerName( ) const ;
Gibt eine Kopie des aufgelösten Warteschlangenmanagernamens (**resolved queue manager name**) zurück.

ImqString resolvedQueueName( ) const ;
Gibt eine Kopie des aufgelösten Warteschlangennamens (**resolved queue name**) zurück.

ImqBoolean syncPointParticipation( ) const ;
Gibt den Wert für die Synchronisationspunktbeteiligung (**syncpoint participation**) zurück, der auf TRUE gesetzt ist, wenn die Optionen (**options**) MQPMO_SYNCPOINT einschließen.

void setSyncPointParticipation(const ImqBoolean *sync*);
Setzt den Wert für die Synchronisationspunktbeteiligung (**syncpoint participation**). Wenn *sync* auf TRUE gesetzt ist, werden die Optionen (**options**) so geändert, dass sie MQPMO_SYNCPOINT einschließen und MQPMO_NO_SYNCPOINT ausschließen. Wenn *sync* auf FALSE gesetzt ist, werden die Optionen (**options**) so geändert, dass sie MQPMO_NO_SYNCPOINT einschließen und MQPMO_SYNCPOINT ausschließen.

Objektdaten (geschützt)

MQPMO *omqpmo*
Die MQPMO-Datenstruktur.

Ursachencodes

- MQRC_STORAGE_NOT_AVAILABLE

C++-Klasse "ImqQueue"

Diese Klasse bindet eine Nachrichtenwarteschlange (ein WebSphere MQ-Objekt des Typs MQOT_Q) ein.

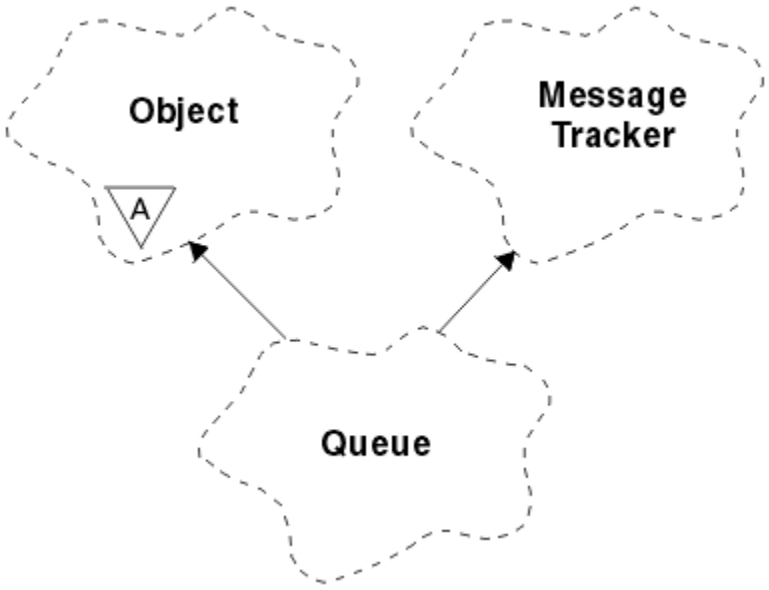


Abbildung 64. ImqQueue, Klasse

Diese Klasse bezieht sich auf die im Abschnitt Tabelle 612 auf Seite 1352 aufgelisteten MQI-Aufrufe.

Objektattribute

backout requeue name
Name der Warteschlange zum Wiedereinreihen überzähliger zurückgesetzter Nachrichten. Dieses Attribut ist schreibgeschützt.

backout threshold
Zurückstellungsschwellenwert. Dieses Attribut ist schreibgeschützt.

base queue name
Name der Warteschlange, in den der Aliasname aufgelöst wird. Dieses Attribut ist schreibgeschützt.

Clustername
Clustername. Dieses Attribut ist schreibgeschützt.

cluster namelist name
Name der Clusternamensliste. Dieses Attribut ist schreibgeschützt.

cluster workload rank
Rangordnung der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload priority
Priorität der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload use queue
Wert der Warteschlange zur Verwendung der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

creation date
Erstellungsdatum der Warteschlange. Dieses Attribut ist schreibgeschützt.

creation time
Erstellungszeit der Warteschlange. Dieses Attribut ist schreibgeschützt.

current depth
Anzahl der Nachrichten in der Warteschlange. Dieses Attribut ist schreibgeschützt.

default bind
Standardbindung. Dieses Attribut ist schreibgeschützt.

default input open option
Standardoption für Öffnen für Eingaben. Dieses Attribut ist schreibgeschützt.

default persistence
Standardmäßige Nachrichtenpersistenz. Dieses Attribut ist schreibgeschützt.

default priority
Standardmäßige Nachrichtenpriorität. Dieses Attribut ist schreibgeschützt.

definition type
Typ der Warteschlangendefinition Dieses Attribut ist schreibgeschützt.

depth high event
Steuerattribut für "Warteschlangenlänge hoch"-Ereignisse. Dieses Attribut ist schreibgeschützt.

depth high limit
Oberer Grenzwert für die Warteschlangenlänge. Dieses Attribut ist schreibgeschützt.

depth low event
Steuerattribut für Ereignisse vom Typ "Queue Depth Low" (Warteschlangenlänge niedrig). Dieses Attribut ist schreibgeschützt.

depth low limit
Unterer Grenzwert für die Warteschlangenlänge. Dieses Attribut ist schreibgeschützt.

depth maximum event
Steuerattribut für die maximalen Ereignisse der Warteschlangentiefe. Dieses Attribut ist schreibgeschützt.

distribution list reference
Optionaler Verweis auf eine "ImqDistributionList"-Verteilerliste, der verwendet werden kann, um Nachrichten an mehrere Warteschlangen, einschließlich dieser, zu verteilen. Der Anfangswert ist null.

Anmerkung: Wenn ein "ImqQueue"-Objekt geöffnet wird, werden alle geöffneten "ImqDistributionList"-Objekte, auf die es verweist, automatisch geschlossen.

distribution lists
Die Fähigkeit einer Übertragungswarteschlange zur Unterstützung von Verteilerlisten. Dieses Attribut ist schreibgeschützt.

dynamic queue name
Name der dynamischen Warteschlangen. Der Anfangswert ist AMQ.* Für alle Windows-, UNIX-und Linux -Plattformen.

harden get backout
Gibt an, ob der Rücksetzungszähler permanent gespeichert werden soll. Dieses Attribut ist schreibgeschützt.

index type
Indextyp. Dieses Attribut ist schreibgeschützt.

inhibit get
Gibt an, ob GET-Operationen zulässig sind. Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für Aliaswarteschlangen oder lokale Warteschlangen gültig.

inhibit put
Gibt an, ob PUT-Operationen zulässig sind. Der Anfangswert hängt von der Warteschlangendefinition ab.

initiation queue name
Name der Initialisierungswarteschlange. Dieses Attribut ist schreibgeschützt.

maximum depth
Maximal zulässige Nachrichtenanzahl für die Warteschlange. Dieses Attribut ist schreibgeschützt.

maximum message length
Maximale Länge für alle Nachrichten in dieser Warteschlange; diese kann kleiner sein als die maximale Länge für eine beliebige von dem zugeordneten Warteschlangenmanager verwaltete Warteschlange. Dieses Attribut ist schreibgeschützt.

message delivery sequence
Gibt an, ob Nachrichtenpriorität relevant ist. Dieses Attribut ist schreibgeschützt.

next distributed queue
Das nächste Objekt dieser Klasse in nicht festgelegter Reihenfolge, das denselben Verteilerlistenverweis (**distribution list reference**) wie dieses Objekt aufweist. Der Anfangswert ist null.

Wenn ein Objekt in einer Kette gelöscht wird, werden das vorherige und das nächste Objekt aktualisiert, sodass ihre verteilten Warteschlangenverbindungen nicht mehr auf das gelöschte Objekt zeigen.

non-persistent message class
Zuverlässigkeitsstufe für nicht persistente Nachrichten, die in diese Warteschlange eingereiht wurden. Dieses Attribut ist schreibgeschützt.

open input count
Anzahl der "ImqQueue"-Objekte, die für die Eingabe geöffnet sind. Dieses Attribut ist schreibgeschützt.

open output count
Anzahl der "ImqQueue"-Objekte, die für die Ausgabe geöffnet sind. Dieses Attribut ist schreibgeschützt.

previous distributed queue
Das vorherige Objekt dieser Klasse in nicht festgelegter Reihenfolge, das denselben Verteilerlistenverweis (**distribution list reference**) wie dieses Objekt aufweist. Der Anfangswert ist null.

Wenn ein Objekt in einer Kette gelöscht wird, werden das vorherige und das nächste Objekt aktualisiert, sodass ihre verteilten Warteschlangenverbindungen nicht mehr auf das gelöschte Objekt zeigen.

process name
Name der Prozessdefinition. Dieses Attribut ist schreibgeschützt.

queue accounting
Stufe der Abrechnungsdaten für Warteschlangen. Dieses Attribut ist schreibgeschützt.

Name des Warteschlangenmanagers
Name des (möglicherweise fernen) Warteschlangenmanagers, auf dem die Warteschlange sich befindet. Verwechseln Sie den hier benannten Warteschlangenmanager nicht mit dem Verbindungsverweis

(**connection reference**) von "ImqObject", der auf den (lokalen) Warteschlangenmanager verweist, der eine Verbindung bereitstellt. Der Anfangswert ist null.

queue monitoring
Stufe der Überwachungsdatenerfassung für die Warteschlange. Dieses Attribut ist schreibgeschützt.

queue statistics
Stufe der statistischen Daten für die Warteschlange. Dieses Attribut ist schreibgeschützt.

Warteschlangentyp
Warteschlangentyp. Dieses Attribut ist schreibgeschützt.

Ferner Warteschlangenmanagername
Name des fernen Warteschlangenmanagers. Dieses Attribut ist schreibgeschützt.

remote queue name
Name der fernen Warteschlange, unter dem diese auf dem fernen Warteschlangenmanager bekannt ist. Dieses Attribut ist schreibgeschützt.

resolved queue manager name
Aufgelöster Warteschlangenmanagername. Dieses Attribut ist schreibgeschützt.

resolved queue name
Aufgelöster Warteschlangenname. Dieses Attribut ist schreibgeschützt.

retention interval
Warteschlangensicherungsintervall. Dieses Attribut ist schreibgeschützt.

scope
Geltungsbereich der Warteschlangendefinition. Dieses Attribut ist schreibgeschützt.

service interval
Serviceintervall. Dieses Attribut ist schreibgeschützt.

service interval event
Steuerattribut für Serviceintervallereignisse. Dieses Attribut ist schreibgeschützt.

shareability
Gibt an, ob die Warteschlange gemeinsam genutzt werden kann. Dieses Attribut ist schreibgeschützt.

Speicherklasse
Speicherklasse. Dieses Attribut ist schreibgeschützt.

transmission queue name
Name der Übertragungswarteschlange. Dieses Attribut ist schreibgeschützt.

trigger control
Auslösesteuerung. Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für eine lokale Warteschlange gültig.

trigger data
Auslösedaten Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für eine lokale Warteschlange gültig.

trigger depth
Auslöserschwelle Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für eine lokale Warteschlange gültig.

trigger message priority
Nachrichtenprioritätsschwelle für Auslöser Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für eine lokale Warteschlange gültig.

trigger type
Auslösertyp Der Anfangswert hängt von der Warteschlangendefinition ab. Dieses Attribut ist nur für eine lokale Warteschlange gültig.

usage
Verwendung. Dieses Attribut ist schreibgeschützt.

Konstruktoren

ImqQueue( );
Der Standardkonstruktor.

ImqQueue(const ImqQueue & *Warteschlange*);
Der Kopierkonstruktor. Der Öffnungsstatus (**open status**) von "ImqObject" ist dann auf FALSE gesetzt.

ImqQueue(const char * *name*);
Legt den Namen (**name**) von "ImqObject" fest.

Objektmethoden (öffentlich)

void operator = (const ImqQueue & *Warteschlange*);
Führt bei Bedarf einen Schließvorgang aus und kopiert anschließend Instanzdaten aus *queue*. Der Öffnungsstatus (**open status**) von "ImqObject" ist dann auf FALSE gesetzt.

ImqBoolean backoutRequeueName(ImqString & *Name*);
Stellt eine Kopie des Namens der Warteschlange zum Wiedereinreihen zurückgesetzter Nachrichten (**backout requeue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString backoutRequeueName( );
Gibt den Namen der Warteschlange zum Wiedereinreihen zurückgesetzter Nachrichten (**backout requeue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean backoutThreshold(MQLONG & *schwellenwert*);
Stellt eine Kopie des Rücksetzschwellenwerts (**backout threshold**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG backoutThreshold( );
Gibt den Rücksetzschwellenwert (**backout threshold**) ohne Angabe möglicher Fehler zurück.

ImqBoolean baseQueueName(ImqString & *Name*);
Gibt eine Kopie des Basiswarteschlangennamens (**base queue name**) zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString baseQueueName( );
Gibt den Basiswarteschlangennamen (**base queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterName(ImqString & *Name*);
Stellt eine Kopie des Clusternamens (**cluster name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString clusterName( );
Gibt den Clusternamen (**cluster name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterNamelistName (ImqString & *Name*);
Stellt eine Kopie des Namens der Clusternamensliste (**cluster namelist name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString clusterNamelistName( );
Gibt den Namen der Clusternamensliste (**cluster namelist name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);
Stellt eine Kopie des Prioritätswerts für die Clusterauslastung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkLoadPriority ( );
Gibt den Wert für die Priorität der Clusterauslastung ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkLoadRank (MQLONG & rank);
Stellt eine Kopie des Werts für die Rangfolge der Clusterauslastung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkLoadRank ( );
Gibt den Wert für die Rangfolge der Clusterauslastung ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Stellt eine Kopie des Werts der Warteschlange zur Verwendung der Clusterauslastung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkLoadUseQ ( );
Gibt den Wert der Warteschlange zur Verwendung der Clusterauslastung ohne Angabe möglicher Fehler zurück.

ImqBoolean creationDate(ImqString & ***Datum*** **);**
Stellt eine Kopie des Erstellungsdatums (**creation date**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString creationDate( );
Gibt das Erstellungsdatum (**creation date**) Angabe möglicher Fehler zurück.

ImqBoolean creationTime(ImqString & ***Zeit*** **);**
Stellt eine Kopie der Erstellungszeit (**creation time**) zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString creationTime( );
Gibt die Erstellungszeit (**creation time**) ohne Angabe möglicher Fehler zurück.

ImqBoolean currentDepth(MQLONG & ***Tiefe*** **);**
Gibt eine Kopie der aktuellen Länge (**current depth**) zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG currentDepth( );
Gibt die aktuelle Länge (**current depth**) ohne Angabe möglicher Fehler zurück.

ImqBoolean defaultInputOpenOption(MQLONG & ***Option*** **);**
Stellt eine Kopie der Standardoption für die Öffnung zur Eingabe (**default input open option**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG defaultInputOpenOption( );
Gibt die Standardoption für die Öffnung zur Eingabe (**default input open option**) ohne Angabe möglicher Fehler zurück.

ImqBoolean defaultPersistence(MQLONG & ***persistenz*** **)**
Stellt eine Kopie der Standardpersistenz (**default persistence**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG defaultPersistence( );
Gibt die Standardpersistenz (**default persistence**) ohne Angabe möglicher Fehler zurück.

ImqBoolean defaultPriority(MQLONG & ***priorität*** **);**
Stellt eine Kopie der Standardpriorität (**default priority**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG defaultPriority( );
Gibt die Standardpriorität (**default priority**) ohne Angabe möglicher Fehler zurück.

ImqBoolean defaultBind(MQLONG & ***bind*** **);**
Stellt eine Kopie der Standardbindung (**default bind**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG defaultBind( );
Gibt die Standardbindung (**default bind**) ohne Angabe möglicher Fehler zurück.

ImqBoolean definitionType(MQLONG & ***typ*** **);**
Stellt eine Kopie des Definitionstyps (**definition type**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG definitionType( );
Gibt den Definitionstyp (**definition type**) ohne Angabe möglicher Fehler zurück.

ImqBoolean depthHighEreignis(MQLONG & ***Ereignis*** **);**
Stellt eine Kopie des Aktivierungsstatus des Ereignisses des Typs **depth high event** (Warteschlangenlänge hoch) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG depthHighEvent( );
Gibt den Aktivierungsstatus des Ereignisses des Typs **depth high event** (Warteschlangenlänge hoch) ohne Angabe möglicher Fehler zurück.

ImqBoolean depthHighLimit(MQLONG &*limit*);
Stellt eine Kopie des Grenzwerts für 'Warteschlangenlänge hoch' (**depth high limit**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG depthHighLimit( );
Gibt den Grenzwert für 'Warteschlangenlänge hoch' (**depth high limit**) ohne Angabe möglicher Fehler zurück.

ImqBoolean depthLowEvent(MQLONG &*event*);
Stellt eine Kopie des Aktivierungsstatus des Ereignisses des Typs **depth low event** (Warteschlangenlänge niedrig) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG depthLowEvent( );
Gibt den Aktivierungsstatus des Ereignisses des Typs **depth low event** (Warteschlangenlänge niedrig) ohne Angabe möglicher Fehler zurück.

ImqBoolean depthLowLimit(MQLONG &*limit*)
Stellt eine Kopie des Grenzwerts für 'Warteschlangenlänge niedrig' (**depth low limit**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG depthLowLimit( );
Gibt den Grenzwert für 'Warteschlangenlänge niedrig' (**depth low limit**) ohne Angabe möglicher Fehler zurück.

ImqBoolean depthMaximumEvent(MQLONG &*event*);
Stellt eine Kopie des Aktivierungsstatus des Ereignisses des Typs **depth maximum event** (Warteschlangenlänge maximal) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG depthMaximumEvent( );
Gibt den Aktivierungsstatus des Ereignisses des Typs **depth maximum event** (Warteschlangenlänge maximal) ohne Angabe möglicher Fehler zurück.

ImqDistributionList * distributionListReference( ) const ;
Gibt den Verteilerlistenverweis (**distribution list reference**) zurück.

void setDistributionListReference(ImqDistribution-Liste &*Liste*)
Legt den Verteilerlistenverweis (**distribution list reference**) fest.

void setDistributionListReference(ImqDistributionList ** *list* **= 0);
Legt den Verteilerlistenverweis (**distribution list reference**) fest.

ImqBoolean distributionLists(Unterstützung für MQLONG &)
Stellt eine Kopie des Werts für die Verteilerlisten (**distribution lists**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG distributionLists( );
Gibt den Wert für die Verteilerlisten (**distribution lists**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setDistributionLists(const MQLONG *support*);
Setzt den Wert für die Verteilerlisten (**distribution lists**). Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString dynamicQueueName( ) const ;
Gibt eine Kopie des Namens der dynamischen Warteschlange (**dynamic queue name**) zurück.

ImqBoolean setDynamicQueueName(const char ** *name* **);
Legt den Namen der dynamischen Warteschlange (**dynamic queue name**) fest. Der Name der dynamischen Warteschlange (**dynamic queue name**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) von "ImqObject" auf FALSE gesetzt ist. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean get(ImqMessage &*msg*, ImqGetMessageOptions &*Optionen*);
Ruft eine Nachricht aus der Warteschlange ab und verwendet dabei den angegebenen Wert für *options*. Ruft bei Bedarf die "ImqObject"-Methode **openFor** auf, um sicherzustellen, dass die Optionen zum Öffnen (**open options**) von "ImqObject" einen der MQOO_INPUT_*-Werte oder den Wert

MQOO_BROWSE einschließen, in Abhängigkeit von dem Wert für *options*. Wenn das *msg*-Objekt einen automatischen Puffer (**automatic buffer**) für "ImqCache" aufweist, vergrößert sich der Speicher, um alle abgerufene Nachrichten aufnehmen zu können. Die Methode **clearMessage** wird vor dem Abruf für das *msg*-Objekt aufgerufen.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Anmerkung: Das Ergebnis des Methodenaufrufs ist FALSCH, wenn das ImqObject **Ursachencode** MQRC_TRUNCATED_MSG_FAILED ist, auch wenn dieser **Ursachencode** als Warnung klassifiziert wird. Wenn eine abgeschnittene Nachricht akzeptiert wird, spiegelt die Nachrichtenlänge (**message length**) von "ImqCache" die abgeschnittene Länge wider. Bei jedem Ereignis gibt die Gesamtlänge der Nachricht (**total message length**) von "ImqMessage" die Byteanzahl an, die verfügbar war.

ImqBoolean get(ImqMessage & *Nachricht*);
Es gilt dasselbe wie für die vorherige Methode, außer dass Standardnachrichtenabrufoptionen verwendet werden.

ImqBoolean get(ImqMessage & *msg*, ImqGetMessageOptions & *Optionen*, const size_t *puffergröße*);
Es gilt dasselbe wie für die vorherigen beiden Methode, außer dass ein Überschreiben für *buffer-size* angegeben wird. Wenn das *msg*-Objekt einen automatischen Puffer (**automatic buffer**) für "ImqCache" einsetzt, wird die Methode **resizeBuffer** vor dem Nachrichtenabruf für das *msg*-Objekt aufgerufen und der Puffer vergrößert sich nicht weiter, um längere Nachrichten aufnehmen zu können.

ImqBoolean get(ImqMessage & *msg*, const size_t *puffergröße*);
Es gilt dasselbe wie für die vorherige Methode, außer dass Standardnachrichtenabrufoptionen verwendet werden.

ImqBoolean hardenGetBackout(MQLONG & *permanent*);
Stellt eine Kopie des Werts für die Anzahl der vor der Zurückstellung aufzuzeichnenden Versuche (**harden get backout**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG hardenGetBackout( );
Gibt den Wert für die Anzahl der vor der Zurückstellung aufzuzeichnenden Versuche (**harden get backout**) ohne Angabe möglicher Fehler zurück.

ImqBoolean indexType(MQLONG & *typ*);
Stellt eine Kopie des Indextyps (**index type**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG indexType( );
Gibt den Indextyp (**index type**) ohne Angabe möglicher Fehler zurück.

ImqBoolean inhibitGet(MQLONG & *sperren*);
Stellt eine Kopie des Werts für das Sperren von GET-Operationen (**inhibit get**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG inhibitGet( );
Gibt den Wert für das Sperren von GET-Operationen (**inhibit get**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setInhibitGet(const MQLONG *inhibit*);
Legt den Wert für das Sperren von GET-Operationen (**inhibit get**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean inhibitPut(MQLONG & *inhibieren*);
Stellt eine Kopie des Werts für das Sperren von PUT-Operationen (**inhibit put**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG inhibitPut( );
Gibt den Wert für das Sperren von PUT-Operationen (**inhibit put**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setInhibitPut(const MQLONG *inhibit*);
Legt den Wert für das Sperren von PUT-Operationen (**inhibit put**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean initiationQueueName(ImqString & ***Name*** **);**
Gibt eine Kopie des Namens der Initialisierungswarteschlange (**initiation queue name**) zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString initiationQueueName( );
Gibt den Namen der Initialisierungswarteschlange (**initiation queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumDepth(MQLONG & ***Tiefe*** **);**
Stellt eine Kopie der maximalen Länge (**maximum depth**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumDepth( );
Gibt die maximale Länge (**maximum depth**) ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumMessageLength(MQLONG & ***length*** **)**
Stellt eine Kopie der maximalen Nachrichtenlänge (**maximum message length**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumMessageLength( );
Gibt die maximale Nachrichtenlänge (**maximum message**) ohne Angabe möglicher Fehler zurück.

ImqBoolean messageDeliveryFolge(MQLONG & ***Folge*** **);**
Stellt eine Kopie der Reihenfolge bei der Nachrichtenübertragung (**message delivery sequence**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG messageDeliverySequence( );
Gibt den Wert für die Reihenfolge bei der Nachrichtenübertragung (**message delivery sequence**) ohne Angabe möglicher Fehler zurück.

ImqQueue * nextDistributedQueue( ) const ;
Gibt die nächste verteilte Warteschlange (**next distributed queue**) zurück.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);
Stellt eine Kopie der Klasse für nicht persistente Nachrichten bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG nonPersistentMessageClass ( );
Gibt den Wert für die Klasse für nicht persistente Nachrichten ohne Angabe möglicher Fehler zurück.

ImqBoolean openInputCount(MQLONG & ***count*** **);**
Stellt eine Kopie der Anzahl der Öffnungen zur Eingabe (**open input count**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG openInputCount( );
Gibt die Anzahl der Öffnungen zur Eingabe (**open count**) ohne Angabe möglicher Fehler zurück.

ImqBoolean openOutputAnzahl(MQLONG & ***Anzahl*** **);**
Stellt eine Kopie der Anzahl der Öffnungen zur Ausgabe (**open output count**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG openOutputCount( );
Gibt die Anzahl der Öffnungen zur Ausgabe (**open output count**) ohne Angabe möglicher Fehler zurück.

ImqQueue * previousDistributedQueue( ) const ;
Gibt die vorherige verteilte Warteschlange (**previous distributed queue**) zurück.

ImqBoolean processName(ImqString & ***Name*** **);**
Stellt eine Kopie des Prozessnamens (**process name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString processName( );
Gibt den Prozessnamen (**process name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean put(ImqMessage & ***Nachricht*** **);**
Reiht eine Nachricht in die Warteschlange ein und verwendet dabei Standardoptionen zum Einreihen von Nachrichten. Verwendet bei Bedarf die "ImqObject"-Methode **openFor**, um sicherzustellen, dass die Optionen zum Öffnen (**open options**) von "ImqObject" MQOO_OUTPUT einschließen.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean put(ImqMessage & *msg*, ImqPutMessageOptions & *pmo*);
Reiht eine Nachricht in die Warteschlange ein und verwendet dabei den angegebenen Wert für *pmo*. Verwendet die Methode ImqObject **openFor** nach Bedarf, um sicherzustellen, dass die ImqObject **-Öffnungsoptionen** MQOO_OUTPUT enthalten und (wenn die *pmo* **Optionen** MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT oder MQPMO_SET_ALL_CONTEXT enthalten) entsprechende MQOO_ * _CONTEXT-Werte.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Anmerkung: Wenn *pmo* einen Kontextverweis (**context reference**) beinhaltet, wird das Referenzobjekt ggf. geöffnet, um einen Kontext bereitzustellen.

ImqBoolean queueAccounting (MQLONG & acctq);
Stellt eine Kopie des Werts für die Warteschlangenabrechnung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueAccounting ( );
Gibt den Wert für die Warteschlangenabrechnung ohne Angabe möglicher Fehler zurück.

ImqString queueManagerName( ) const ;
Gibt den Warteschlangenmanagernamen (**queue manager name**) zurück.

ImqBoolean setQueueManagerName(const char * *name*);
Legt den Warteschlangenmanagernamen (**queue manager name**) fest. Der Warteschlangenmanagername (**queue manager name**) kann nur festgelegt werden, wenn der Öffnungsstatus (**open status**) von "ImqObject" auf FALSE gesetzt ist. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean queueMonitoring (MQLONG & monq);
Stellt eine Kopie des Werts für die Warteschlangenüberwachung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueMonitoring ( );
Gibt den Wert für die Warteschlangenüberwachung ohne Angabe möglicher Fehler zurück.

ImqBoolean queueStatistics (MQLONG & statq);
Stellt eine Kopie des Werts für die Warteschlangenstatistik bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueStatistics ( );
Gibt den Wert für die Warteschlangenstatistik ohne Angabe möglicher Fehler zurück.

ImqBoolean queueType(MQLONG & *typ*);
Stellt eine Kopie des Werts für den Warteschlangentyp (**queue type**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueType( );
Gibt den Warteschlangentyp (**queue type**) ohne Angabe möglicher Fehler zurück.

ImqBoolean remoteQueueManagerName(ImqString & *name*);
Stellt eine Kopie des Namens des fernen Warteschlangenmanagers (**remote queue manager name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString remoteQueueManagerName( );
Gibt den Namen des fernen Warteschlangenmanagers (**remote queue manager name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean remoteQueueName(ImqString & *Name*);
Stellt eine Kopie des Namens der fernen Warteschlange (**remote queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString remoteQueueName( );
Gibt den Namen der fernen Warteschlange (**remote queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean resolvedQueueManagerName(ImqString & *Name*);
Stellt eine Kopie des aufgelösten Warteschlangenmanagernamens (**resolved queue manager name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Anmerkung: Diese Methode schlägt fehl, wenn die Optionen zum Öffnen (**open options**) nicht MQOO_RESOLVE_NAMES enthalten.

ImqString resolvedQueueManagerName( ) ;
Gibt den aufgelösten Warteschlangenmanagernamen (**resolved queue manager name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean resolvedQueueName (ImqString & *Name*);
Stellt eine Kopie des aufgelösten Warteschlangennamens (**resolved queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Anmerkung: Diese Methode schlägt fehl, wenn die Optionen zum Öffnen (**open options**) nicht MQOO_RESOLVE_NAMES enthalten.

ImqString resolvedQueueName( ) ;
Gibt den aufgelösten Warteschlangennamen (**resolved queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean retentionInterval(MQLONG & *intervall*);
Stellt eine Kopie des Aufbewahrungsintervalls (**retention interval**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG retentionInterval( );
Gibt das Aufbewahrungsintervall (**retention interval**) ohne Angabe möglicher Fehler zurück.

ImqBoolean scope(MQLONG & *scope*)
Stellt eine Kopie des Geltungsbereichs (**scope**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG scope( );
Gibt den Geltungsbereich (**scope**) ohne Angabe möglicher Fehler zurück.

ImqBoolean serviceInterval(MQLONG & *intervall*);
Stellt eine Kopie des **Serviceintervall** zur Verfügung. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG serviceInterval( );
Gibt das **Serviceintervall** ohne Angabe möglicher Fehler zurück.

ImqBoolean serviceIntervalEvent(MQLONG & *event*)
Stellt eine Kopie des Aktivierungsstatus des Serviceintervallereignisses (**service interval event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG serviceIntervalEvent( );
Gibt den Aktivierungsstatus des Serviceintervallereignisses (**service interval event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean shareability(MQLONG & *shareability*);
Stellt eine Kopie des Werts für die gemeinsame Nutzung (**shareability**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG shareability( );
Gibt den Wert für die gemeinsame Nutzung (**shareability**) ohne Angabe möglicher Fehler zurück.

ImqBoolean storageClass(ImqString & *Klasse*);
Stellt eine Kopie der Speicherklasse (**storage class**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString storageClass( );
Gibt die Speicherklasse (**storage class**) ohne Angabe möglicher Fehler zurück.

ImqBoolean transmissionQueueName(ImqString & *Name*);
Stellt eine Kopie des Namens der Übertragungswarteschlange (**transmission queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString transmissionQueueName( );
Gibt den Namen der Übertragungswarteschlange (**transmission queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean triggerControl(MQLONG & ***control*** **);**
Stellt eine Kopie des Werts für die Auslösersteuerung (**trigger control**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG triggerControl( );
Gibt den Wert für die Auslösersteuerung (**trigger control**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setTriggerControl(const MQLONG ***control*** **);**
Legt den Wert für die Auslösersteuerung (**trigger control**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean triggerData(ImqString & ***Daten*** **);**
Stellt eine Kopie der Auslöserdaten (**trigger data**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString triggerData( );
Gibt eine Kopie der Auslöserdaten (**trigger data**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setTriggerData(const char * ***data*** **);**
Legt die Auslöserdaten (**trigger data**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean triggerDepth(MQLONG & ***Tiefe*** **);**
Stellt eine Kopie der Auslösertiefe (**trigger depth**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG triggerDepth( );
Gibt die Auslösertiefe (**trigger depth**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setTriggerDepth(const MQLONG ***depth*** **);**
Legt die Auslösertiefe (**trigger depth**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean triggerMessagePriorität(MQLONG & ***priority*** **);**
Stellt eine Kopie der Priorität der Auslösenachricht (**trigger message priority**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG triggerMessagePriority( );
Gibt die Priorität der Auslösenachricht (**trigger message priority**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setTriggerMessagePriority(const MQLONG ***priority*** **);**
Legt die Priorität der Auslösenachricht (**trigger message priority**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean triggerType(MQLONG & ***typ*** **);**
Stellt eine Kopie des Auslösertyps (**trigger type**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG triggerType( );
Gibt den Auslösertyp (**trigger type**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setTriggerType(const MQLONG ***type*** **);**
Legt den Auslösertyp (**trigger type**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean usage(MQLONG & ***usage*** **);**
Stellt eine Kopie des Werts für die Verwendung (**usage**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG usage( );
Gibt den Wert für die Verwendung (**usage**) ohne Angabe möglicher Fehler zurück.

Objektmethoden (geschützt)

void setNextDistributedQueue(ImqQueue * ***queue*** **= 0);**
Legt die nächste verteilte Warteschlange (**next distributed queue**) fest.

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass die Liste der verteilten Warteschlangen dadurch nicht unterbrochen wird.

void setPreviousDistributedQueue(ImqQueue * *queue* = 0);
Legt die vorherige verteilte Warteschlange (**previous distributed queue**) fest.

Achtung: Verwenden Sie diese Funktion nur, wenn Sie sicher sind, dass die Liste der verteilten Warteschlangen dadurch nicht unterbrochen wird.

Ursachencodes

- 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
- (Ursachencodes von MQGET)
- (Ursachencodes von MQPUT)

C++-Klasse "ImqQueueManager"

Diese Klasse bindet einen Warteschlangenmanager (ein WebSphere MQ-Objekt des Typs MQOT_Q_MGR) ein.

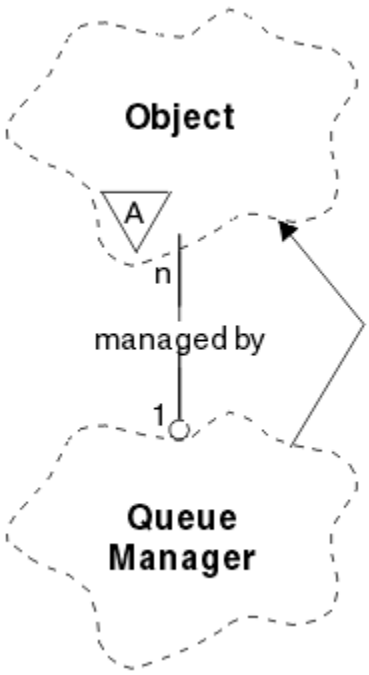


Abbildung 65. ImqQueueManager, Klasse

Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqQueueManager” auf Seite 1354 aufgelisteten MQI-Aufrufe. Nicht alle der aufgelisteten Methoden sind auf alle Plattformen anwendbar; weitere Informationen finden Sie unter ALTER QMGR.

- „Klassenattribute” auf Seite 1428
- „Objektattribute” auf Seite 1428

Klassenattribute

behavior
Steuert das Verhalten von implizitem Verbindungsaufbau und Verbindungsabbau.

IMQ_EXPL_DISC_BACKOUT (0L)
Ein expliziter Aufruf an die Methode **disconnect** bedeutet eine Rücksetzung. Dieses Attribut und IMQ_EXPL_DISC_COMMIT schließen sich gegenseitig aus.

IMQ_EXPL_DISC_COMMIT (1L)
Ein expliziter Aufruf an die Methode **disconnect** bedeutet eine Festschreibung (Standardeinstellung). Dieses Attribut und IMQ_EXPL_DISC_BACKOUT schließen sich gegenseitig aus.

IMQ_IMPL_CONN (2L)
Ein impliziter Verbindungsaufbau ist zulässig (Standardeinstellung).

IMQ_IMPL_DISC_BACKOUT (0L)
Ein impliziter Aufruf an die Methode **disconnect**, der während einer Objektvernichtung auftreten kann, bedeutet eine Rücksetzung. Dieses Attribut und IMQ_IMPL_DISC_COMMIT schließen sich gegenseitig aus.

IMQ_IMPL_DISC_COMMIT (4L)
Ein impliziter Aufruf an die Methode **disconnect**, der während einer Objektvernichtung auftreten kann, bedeutet eine Festschreibung (Standardeinstellung). Dieses Attribut und IMQ_IMPL_DISC_BACKOUT schließen sich gegenseitig aus.

In WebSphere MQ V7.0 und höher müssen C++-Anwendungen, die einen impliziten Verbindungsaufbau verwenden, IMQ_IMPL_CONN zusammen mit jeder anderen Option angeben, die in der Methode `setBehavior()` für ein Objekt der Klasse `ImqQueueManager` übergeben wird. Wenn in Ihrer Anwendung nicht die Methode `setBehavior()` verwendet wird, um die Verhaltensoptionen explizit festzulegen, z. B.

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

, hat dies keine Auswirkung für Sie, da MQ_IMPL_CONN standardmäßig aktiviert ist.

Wenn in Ihrer Anwendung die Verhaltensoptionen explizit festgelegt werden, z. B.

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

, müssen Sie IMQ_IMPL_CONN wie folgt in der Methode `setBehavior()` angeben, damit die Anwendung einen impliziten Verbindungsaufbau durchführen kann:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Objektattribute

accounting connections override
Ermöglicht es Anwendungen, die Einstellung der Werte für die MQI-Abrechnung und die Warteschlangenabrechnung zu überschreiben. Dieses Attribut ist schreibgeschützt.

accounting interval
Das Intervall, in dem temporäre Abrechnungssätze geschrieben werden (in Sekunden). Dieses Attribut ist schreibgeschützt.

activity recording
Steuert die Erzeugung von Aktivitätenberichten. Dieses Attribut ist schreibgeschützt.

adopt new mca check
Die Elemente, die überprüft werden, um festzustellen, ob ein Nachrichtenkanalagent (MCA = Message Channel Agent) übernommen werden soll, wenn ein neuer eingehender Kanal mit demselben Namen wie ein bereits aktiver Nachrichtenkanalagent erkannt wird. Dieses Attribut ist schreibgeschützt.

adopt new mca type
Gibt an, ob die verwaiste Instanz eines Nachrichtenkanalagenten eines bestimmten Kanaltyps automatisch erneut gestartet werden soll, wenn eine neu eingehende Kanalanforderung erkannt wird, die den Parametern zur Prüfung der Übernahme des neuen Nachrichtenkanalagenten entspricht. Dieses Attribut ist schreibgeschützt.

authentication type
Gibt den Authentifizierungstyp an, der ausgeführt wird.

authority event
Steuert Berechtigungsereignisse. Dieses Attribut ist schreibgeschützt.

begin options
Optionen für die Methode **begin**. Der Anfangswert ist MQBO_NONE.

bridge event
Gibt an, ob IMS Bridge-Ereignisse generiert werden sollen. Dieses Attribut ist schreibgeschützt.

channel auto definition
Wert für die automatische Kanaldefinition. Dieses Attribut ist schreibgeschützt.

channel auto definition event
Wert für das Ereignis 'Automatische Kanaldefinition'. Dieses Attribut ist schreibgeschützt.

channel auto definition exit
Exitname für die automatische Kanaldefinition. Dieses Attribut ist schreibgeschützt.

channel event
Gibt an, ob Kanalereignisse generiert werden sollen. Dieses Attribut ist schreibgeschützt.

channel initiator adapters
Die Anzahl der Adaptersubtasks, die für die Verarbeitung von WebSphere MQ-Aufrufen verwendet werden sollen. Dieses Attribut ist schreibgeschützt.

channel initiator control
Gibt an, ob der Kanalinitiator beim Starten des Warteschlangenmanagers automatisch gestartet werden soll. Dieses Attribut ist schreibgeschützt.

channel initiator dispatchers
Die Anzahl an Dispatchern, die für den Kanalinitiator verwendet werden sollen. Dieses Attribut ist schreibgeschützt.

channel initiator trace autostart
Gibt an, ob der Kanalinitiatortrace automatisch gestartet werden soll oder nicht. Dieses Attribut ist schreibgeschützt.

channel initiator trace table size
Die Größe des Tracedatenspeicherbereichs des Kanalinitiators (in MB.) Dieses Attribut ist schreibgeschützt.

channel monitoring
Steuert die Erfassung von Onlineüberwachungsdaten für Kanäle. Dieses Attribut ist schreibgeschützt.

channel reference
Ein Verweis auf eine Kanaldefinition zur Verwendung während der Clientverbindung. Während einer aktiven Verbindung kann dieses Attribut auf null gesetzt werden; es kann jedoch in keinen anderen Wert geändert werden. Der Anfangswert ist null.

Kanalstatistik
Steuert die Erfassung von statistischen Daten für Kanäle. Dieses Attribut ist schreibgeschützt.

character set
ID des codierten Zeichensatzes (CCSID, Coded Character Set Identifier). Dieses Attribut ist schreibgeschützt.

cluster sender monitoring
Steuert die Erfassung von Onlineüberwachungsdaten für automatisch definierte Clustersenderkanäle. Dieses Attribut ist schreibgeschützt.

cluster sender statistics
Steuert die Erfassung statistischer Daten für automatisch definierte Clustersenderkanäle. Dieses Attribut ist schreibgeschützt.

cluster workload data
Daten des Exits für Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload exit
Exitname für Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload length
Länge der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload mru
Wert für die zuletzt verwendeten Kanäle der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

cluster workload use queue
Wert der Warteschlange zur Verwendung der Clusterauslastung. Dieses Attribut ist schreibgeschützt.

command event
Gibt an, ob Befehlsereignisse generiert werden. Dieses Attribut ist schreibgeschützt.

command input queue name
Name der Eingabewarteschlange für Systembefehle. Dieses Attribut ist schreibgeschützt.

command level
Vom Warteschlangenmanager unterstützte Befehlsebene. Dieses Attribut ist schreibgeschützt.

command server control
Gibt an, ob der Befehlsserver beim Starten des Warteschlangenmanagers automatisch gestartet werden soll. Dieses Attribut ist schreibgeschützt.

connect options
Optionen für die Methode **connect**. Der Anfangswert ist MQCNO_NONE. Die folgenden zusätzlichen Werte sind je nach Plattform möglich:

- 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

connection id
Eine eindeutige Kennung, die MQ die zuverlässige Bestimmung einer Anwendung ermöglicht.

connection status
Bei aktiver Verbindung zum Warteschlangenmanager auf TRUE gesetzt. Dieses Attribut ist schreibgeschützt.

connection tag
Ein Tag, das einer Verbindung zugeordnet werden soll. Dieses Attribut kann nur festgelegt werden, wenn keine Verbindung besteht. Der Anfangswert ist null.

cryptographic hardware
Konfigurationsdetails für Verschlüsselungshardware. Für MQ MQI-Clientverbindungen.

dead-letter queue name
Name der Warteschlange für nicht zustellbare Nachrichten. Dieses Attribut ist schreibgeschützt.

default transmission queue name
Gibt die standardmäßige Übertragungswarteschlange an. Dieses Attribut ist schreibgeschützt.

distribution lists
Fähigkeit des Warteschlangenmanagers, Verteilerlisten zu unterstützen.

dns group
Der Name der Gruppe, zu der das TCP-Empfangsprogramm, das für die Behandlung eingehender Übertragungen für die Gruppe mit gemeinsamer Warteschlange zuständig ist, gehören sollte, wenn Unterstützung für Workload Manager Dynamic Domain Name Services verwendet wird. Dieses Attribut ist schreibgeschützt.

dns wlm
Gibt an, ob das TCP-Empfangsprogramm, das eingehende Übertragungen für die Gruppe mit gemeinsamer Warteschlange verarbeitet, mit Workload Manager for Dynamic Domain Name Services registriert werden soll. Dieses Attribut ist schreibgeschützt.

first authentication record
Das erste von mindestens einem Objekt der Klasse "ImqAuthenticationRecord", wobei der Verteilerlistenverweis "ImqAuthenticationRecord" dieses Objekt nicht in einer bestimmten Reihenfolge anspricht. Für MQ MQI-Clientverbindungen.

first managed object
Das erste von mindestens einem Objekt der Klasse "ImqObject", wobei der Verbindungsverweis (**connection reference**) dieses Objekt nicht in einer bestimmten Reihenfolge anspricht. Der Anfangswert ist null.

inhibit event
Steuert Sperrereignisse. Dieses Attribut ist schreibgeschützt.

ip address version
Gibt an, welches IP-Protokoll (IPv4 oder IPv6) für eine Kanalverbindung verwendet werden soll. Dieses Attribut ist schreibgeschützt.

Schlüsselrepository
Position der Schlüsseldatenbankdatei, in der Schlüssel und Zertifikate gespeichert sind. Für WebSphere MQ MQI-Clientverbindungen.

key reset count
Die Anzahl an unverschlüsselten Bytes, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet bzw. empfangen werden. Dieses Attribut gilt nur für Clientverbindungen, die MQCONNX verwenden. Siehe auch ssl key reset count.

listener timer
Das Zeitintervall (in Sekunden) zwischen Versuchen von WebSphere MQ, das Empfangsprogramm erneut zu starten, wenn ein APPC- oder TCP/IP-Fehler aufgetreten ist. Dieses Attribut ist schreibgeschützt.

local event
Steuert lokale Ereignisse. Dieses Attribut ist schreibgeschützt.

logger event
Steuert das Generieren von Ereignissen für das Wiederherstellungsprotokoll. Dieses Attribut ist schreibgeschützt.

lu group name
Der generische LU-Name, den das LU 6.2-Empfangsprogramm für eingehende Transaktionen für eine Gruppe mit gemeinsamer Warteschlange verwenden sollte. Dieses Attribut ist schreibgeschützt.

lu name
Der Name der LU, die für abgehende LU 6.2-Übertragungen verwendet werden soll. Dieses Attribut ist schreibgeschützt.

lu62 arm suffix
Das Suffix des SYS1.PARMLIB-Members APPCPMxx, das die LUADD für diesen Kanalinitiator benennt. Dieses Attribut ist schreibgeschützt.

lu62 channels
Die maximale Anzahl aktiver Kanäle bzw. verbundener Clients, die das Übertragungsprotokoll LU 6.2 verwenden. Dieses Attribut ist schreibgeschützt.

maximum active channels
Die Anzahl an Kanälen, die maximal gleichzeitig aktiv sein können. Dieses Attribut ist schreibgeschützt.

maximum channels
Die maximale Anzahl Kanäle, die gleichzeitig aktiv sein können (einschließlich Serververbindungskanälen mit verbundenen Clients). Dieses Attribut ist schreibgeschützt.

maximum handles
Maximale Anzahl Kennungen. Dieses Attribut ist schreibgeschützt.

maximale Nachrichtenlänge
Maximal mögliche Länge für alle Nachrichten in allen Warteschlangen, die von diesem Warteschlangenmanager verwaltet werden. Dieses Attribut ist schreibgeschützt.

maximum priority
Maximale Nachrichtenpriorität. Dieses Attribut ist schreibgeschützt.

maximum uncommitted messages
Maximale Anzahl nicht festgeschriebener Nachrichten in einer Arbeitseinheit. Dieses Attribut ist schreibgeschützt.

mqi accounting
Steuert die Erfassung von Abrechnungsinformationen für MQI-Daten. Dieses Attribut ist schreibgeschützt.

mqi statistics
Steuert die Erfassung von statistischen Überwachungsdaten für den Warteschlangenmanager. Dieses Attribut ist schreibgeschützt.

outbound port maximum
Der höchste Wert des Portnummernbereichs, der bei der Bindung abgehender Kanäle verwendet werden soll. Dieses Attribut ist schreibgeschützt.

outbound port minimum
Der niedrigste Wert des Portnummernbereichs, der bei der Bindung abgehender Kanäle verwendet werden soll. Dieses Attribut ist schreibgeschützt.

Kennwort
Der Benutzer-ID zugeordnetes Kennwort.

Leistungsereignis
Steuert Leistungsereignisse. Dieses Attribut ist schreibgeschützt.

platform
Plattform, auf der sich der Warteschlangenmanager befindet. Dieses Attribut ist schreibgeschützt.

queue accounting
Steuert die Erfassung von Abrechnungsinformationen für Warteschlangen. Dieses Attribut ist schreibgeschützt.

queue monitoring
Steuert die Erfassung von Onlineüberwachungsdaten für Warteschlangen. Dieses Attribut ist schreibgeschützt.

queue statistics
Steuert die Erfassung von statistischen Daten für Warteschlangen. Dieses Attribut ist schreibgeschützt.

receive timeout
Gibt an, wie lange ein TCP/IP-Nachrichtenkanal ungefähr auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt. Dieses Attribut ist schreibgeschützt.

receive timeout minimum
Die Mindestzeit, die ein TCP/IP-Kanal auf den Empfang von Daten, einschließlich Überwachungssignalen, vom Partner wartet, bevor er in den inaktiven Zustand zurückkehrt. Dieses Attribut ist schreibgeschützt.

receive timeout type
Ein Qualifikationsmerkmal, das für das Empfangszeitlimit (**receive timeout**) gilt. Dieses Attribut ist schreibgeschützt.

remote event
Steuert ferne Ereignisse. Dieses Attribut ist schreibgeschützt.

repository name
Repository-Name. Dieses Attribut ist schreibgeschützt.

repository namelist
Repository-Namenslistenname. Dieses Attribut ist schreibgeschützt.

shared queue manager name
Gibt an, ob MQOPEN-Aufrufe einer gemeinsam genutzten Warteschlange, wobei ObjectQMgrName ein anderer Warteschlangenmanager innerhalb der Gruppe mit gemeinsamer Warteschlange ist, in einen Öffnungsvorgang (open) der gemeinsam genutzten Warteschlange auf dem lokalen Warteschlangenmanager aufgelöst werden sollten. Dieses Attribut ist schreibgeschützt.

ssl event
Gibt an, ob SSL-Ereignisse generiert werden. Dieses Attribut ist schreibgeschützt.

ssl FIPS required
Gibt an, ob für die Verschlüsselungen in WebSphere MQ-Software nur FIPS-zertifizierte Algorithmen verwendet werden sollen. Dieses Attribut ist schreibgeschützt.

ssl key reset count
Die Anzahl an unverschlüsselten Bytes, die vor einer Neuvereinbarung des geheimen Schlüssels in einem SSL-Dialog gesendet bzw. empfangen werden. Dieses Attribut ist schreibgeschützt.

start-stop event
Steuert Start-Stopp-Ereignisse. Dieses Attribut ist schreibgeschützt.

statistics interval
Gibt an, wie oft statistische Überwachungsdaten in die Überwachungswarteschlange geschrieben werden. Dieses Attribut ist schreibgeschützt.

syncpoint availability
Verfügbarkeit der Synchronisationspunktbeteiligung. Dieses Attribut ist schreibgeschützt.

Anmerkung: Vom Warteschlangenmanager koordinierte globale Arbeitseinheiten werden auf der IBM i-Plattform nicht unterstützt.

tcp channels
Die maximale Anzahl aktueller Kanäle oder verbundener Clients, die das Übertragungsprotokoll TCP/IP verwenden. Dieses Attribut ist schreibgeschützt.

tcp keepalive
Gibt an, ob die TCP-Keepalive-Funktion verwendet werden soll, um zu überprüfen, ob das andere Verbindungsende noch verfügbar ist. Dieses Attribut ist schreibgeschützt.

tcp name
Der Name des einzigen bzw. standardmäßigen TCP/IP-Systems (je nach dem Wert für **tcp stack type**), das verwendet werden soll. Dieses Attribut ist schreibgeschützt.

tcp stack type
Gibt an, ob der Kanalinitiator nur den in **tcp name** angegebenen TCP/IP-Adressraum verwenden darf oder ob er eine Bindung zu einer beliebigen ausgewählten TCP/IP-Adresse herstellen kann. Dieses Attribut ist schreibgeschützt.

trace route recording
Steuert die Aufzeichnung von Informationen über die Tracefunktion für Routes. Dieses Attribut ist schreibgeschützt.

trigger interval
Auslöseintervall. Dieses Attribut ist schreibgeschützt.

Benutzer-ID
Auf den Plattformen UNIX and Linux gibt dieses Feld die reale Benutzer-ID der Anwendung an. Auf Windows-Plattformen gibt dieses Feld die Benutzer-ID der Anwendung an.

Konstruktoren

ImqQueueManager( );
Der Standardkonstruktor.

ImqQueueManager (const ImqQueueManager & *Manager*);
Der Kopierkonstruktor. Der Verbindungsstatus (**connection status**) ist dann auf FALSE gesetzt.

ImqQueueManager (const char * *Name*);
Setzt den **Namen** von 'ImqObject' auf *Name*.

Destruktoren

Wenn ein "ImqQueueManager"-Objekt gelöscht wird, wird die Verbindung automatisch getrennt.

Klassenmethoden (öffentlich)

static MQLONG behavior( );
Gibt das Verhalten (**behavior**) zurück.

void setBehavior(const MQLONG *Verhalten* = 0);
Legt das Verhalten (**behavior**) fest.

Objektmethoden (öffentlich)

void operator = (const ImqQueueManager & *mgr*);
Trennt ggf. die Verbindung und kopiert Instanzdaten aus *mgr*. Der Verbindungsstatus (**connection status**) ist dann auf FALSE gesetzt.

ImqBoolean accountingConnOverride (MQLONG & statint);
Stellt eine Kopie des Werts zur Überschreibung der Abrechnungsverbindungen bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG accountingConnOverride ( );
Gibt den Wert zur Überschreibung der Abrechnungsverbindungen ohne Angabe möglicher Fehler zurück.

ImqBoolean accountingInterval (MQLONG & statint);
Stellt eine Kopie des Werts für das Abrechnungsintervall bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG accountingInterval ( );
Gibt den Wert für das Abrechnungsintervall ohne Angabe möglicher Fehler zurück.

ImqBoolean activityRecording (MQLONG & rec);
Stellt eine Kopie des Werts für die Aktivitätsaufzeichnung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG activityRecording ( );
Gibt den Wert für die Aktivitätsaufzeichnung ohne Angabe möglicher Fehler zurück.

ImqBoolean adoptNewMCACheck (MQLONG & check);
Stellt eine Kopie des Werts für die Prüfung der Übernahme des neuen Nachrichtenkanalagenten bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG adoptNewMCACheck ( );
Gibt den Wert für die Prüfung der Übernahme des neuen Nachrichtenkanalagenten ohne Angabe möglicher Fehler zurück.

ImqBoolean adoptNewMCAType (MQLONG & type);
Stellt eine Kopie des Typs der Übernahme des neuen Nachrichtenkanalagenten bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG adoptNewMCAType ( );
Gibt den Typ der Übernahme des neuen Nachrichtenkanalagenten ohne Angabe möglicher Fehler zurück.

QLONG authenticationType ( ) const;
Gibt den Authentifizierungstyp zurück.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);
Legt den Authentifizierungstyp fest.

ImqBoolean authorityEvent(MQLONG & ***ereignis*** **);**
Stellt eine Kopie des Aktivierungsstatus des Berechtigungsereignisses (**authority event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG authorityEvent( );
Gibt den Aktivierungsstatus des Berechtigungsereignisses (**authority event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean backout( );
Setzt nicht festgeschriebene Änderungen zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean begin( );
Startet eine Arbeitseinheit. Die Startoptionen (**begin options**) wirken sich auf das Verhalten dieser Methode aus. Bei erfolgreicher Ausführung wird TRUE zurückgegeben, aber TRUE wird auch dann zurückgegeben, wenn der zugrunde liegende MQBEGIN-Aufruf MQRC_NO_EXTERNAL_PARTICIPANTS oder MQRC_PARTICIPANT_NOT_AVAILABLE zurückgibt (wobei beide MQCC_WARNING zugeordnet sind).

MQLONG beginOptions( ) const ;
Gibt die Startoptionen (**begin options**) zurück.

void setBeginOptionen (const MQLONG ***Optionen*** **= MQBO_NONE);**
Legt die Startoptionen (**begin options**) fest.

ImqBoolean bridgeEvent (MQLONG & event);
Stellt eine Kopie des Werts für das Bridge-Ereignis bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG bridgeEvent ( );
Gibt den Wert für das Bridge-Ereignis ohne Angabe möglicher Fehler zurück.

ImqBoolean channelAutoDefinition (MQLONG & ***wert*** **);**
Stellt eine Kopie des Werts für die automatische Kanaldefinition (**channel auto definition**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelAutoDefinition( );
Gibt den Wert für die automatische Kanaldefinition (**channel auto definition**) ohne Angabe möglicher Fehler zurück.

ImqBoolean channelAutoDefinitionEvent(MQLONG & ***wert*** **);**
Stellt eine Kopie des Werts für das Ereignis 'Automatische Kanaldefinition' (**channel auto definition event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelAutoDefinitionEvent( );
Gibt den Wert für das Ereignis 'Automatische Kanaldefinition' (**channel auto definition event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean channelAutoDefinitionExit(ImqString & ***Name*** **);**
Stellt eine Kopie des Exitnamens für die automatische Kanaldefinition (**channel auto definition exit**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString channelAutoDefinitionExit( );
Gibt den Exitnamen für die automatische Kanaldefinition (**channel auto definition exit**) ohne Angabe möglicher Fehler zurück.

ImqBoolean channelEvent (MQLONG & event);
Stellt eine Kopie des Werts für das Kanalereignis bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelEvent( );
Gibt den Wert für das Kanalereignis ohne Angabe möglicher Fehler zurück.

MQLONG channelInitiatorAdapters ( );
Gibt den Wert für die Kanalinitiatoradapter ohne Angabe möglicher Fehler zurück.

ImqBoolean channelInitiatorAdapters (MQLONG & adapters);
Stellt eine Kopie des Werts für die Kanalinitiatoradapter bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelInitiatorControl ( );
Gibt den Startwert für den Kanalinitiator ohne Angabe möglicher Fehler zurück.

ImqBoolean channelInitiatorControl (MQLONG & init);
Stellt eine Kopie des Startwerts für die Kanalinitiatorsteuerung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelInitiatorDispatchers ( );
Gibt den Wert für die Kanalinitiatordispatcher ohne Angabe möglicher Fehler zurück.

ImqBoolean channelInitiatorDispatchers (MQLONG & dispatchers);
Stellt eine Kopie des Werts für die Kanalinitiatordispatcher bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelInitiatorTraceAutoStart ( );
Gibt den Wert für das automatische Starten des Kanalinitiatortrace ohne Angabe möglicher Fehler zurück.

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);
Stellt eine Kopie des Werts für das automatische Starten des Kanalinitiatortrace bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelInitiatorTraceTableSize ( );
Gibt den Wert für die Tabellengröße des Kanalinitiatortrace ohne Angabe möglicher Fehler zurück.

ImqBoolean channelInitiatorTraceTableSize (MQLONG & size);
Stellt eine Kopie des Werts für die Tabellengröße des Kanalinitiatortrace bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean channelMonitoring (MQLONG & monchl);
Stellt eine Kopie des Werts für die Kanalüberwachung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelMonitoring ( );
Gibt den Wert für die Kanalüberwachung ohne Angabe möglicher Fehler zurück.

ImqBoolean channelReference(ImqChannel * & ***Kanal*** **);**
Stellt eine Kopie des Kanalverweises (**channel reference**) bereit. Wenn der Kanalverweis (**channel reference**) ungültig ist, wird *pchannel* auf null gesetzt. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqChannel * channelReference( );
Gibt den Kanalverweis (**channel reference**) ohne Angabe möglicher Fehler zurück.

ImqBoolean setChannelReference (ImqChannel & ***Kanal*** **)**
Legt den Kanalverweis (**channel reference**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setChannelReference (ImqChannel * ***Kanal*** **= 0);**
Legt den Kanalverweis (**channel reference**) (erneut) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean channelStatistics (MQLONG & statchl);
Stellt eine Kopie des Werts für die Kanalstatistik bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG channelStatistics ( );
Gibt den Wert für die Kanalstatistik ohne Angabe möglicher Fehler zurück.

ImqBoolean characterSet(MQLONG & ***ccsid*** **);**
Stellt eine Kopie des Zeichensatzes (**character set**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG characterSet( );
Gibt eine Kopie des Zeichensatzes (**character set**) ohne Angabe möglicher Fehler zurück.

MQLONG clientSslKeyResetCount ( ) const;
Gibt den Wert für die Anzahl der Rücksetzungen für SSL-Schlüssel für Clientverbindungen zurück.

void setClientSslKeyResetCount(const MQLONG count);
Legt die Anzahl der Rücksetzungen für SSL-Schlüssel für Clientverbindungen fest.

ImqBoolean clusterSenderMonitoring (MQLONG & monacls);
Stellt eine Kopie des Standardwerts für die Clustersenderüberwachung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterSenderMonitoring ( );
Gibt den Standardwert für die Clustersenderüberwachung ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterSenderStatistics (MQLONG & statacls);
Stellt eine Kopie des Werts für die Clustersenderstatistik bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterSenderStatistics ( );
Gibt den Wert für die Clustersenderstatistik ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkloadData (ImqString & ***Daten*** **);**
Stellt eine Kopie der Exitdaten für Clusterauslastung (**cluster workload exit data**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString clusterWorkloadData( );
Gibt die Exitdaten für Clusterauslastung (**cluster workload exit data**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkloadExit (ImqString & ***Name*** **);**
Stellt eine Kopie des Exitnamens für Clusterauslastung (**cluster workload exit name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString clusterWorkloadExit( );
Gibt den Exitnamen für Clusterauslastung (**cluster workload exit name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkloadLength (MQLONG & ***länge*** **);**
Stellt eine Kopie der Clusterauslastungslänge (**cluster workload length**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkloadLength( );
Gibt die Clusterauslastungslänge (**cluster workload length**) ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
Stellt eine Kopie des Werts der für die Clusterauslastung zuletzt verwendeten Kanäle bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkLoadMRU ( );
Gibt den Wert der für die Clusterauslastung zuletzt verwendeten Kanäle ohne Angabe möglicher Fehler zurück.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Stellt eine Kopie des Werts der Warteschlange zur Verwendung der Clusterauslastung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG clusterWorkLoadUseQ ( );
Gibt den Wert der Warteschlange zur Verwendung der Clusterauslastung ohne Angabe möglicher Fehler zurück.

ImqBoolean commandEvent (MQLONG & event);
Stellt eine Kopie des Werts für das Befehlsereignis bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG commandEvent ( );
Gibt den Wert für das Befehlsereignis ohne Angabe möglicher Fehler zurück.

ImqBoolean commandInputQueueName(ImqString &*Name*);
Stellt eine Kopie des Namens der Eingabewarteschlange für Befehle (**command input queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString commandInputQueueName( );
Gibt den Namen der Eingabewarteschlange für Befehle (**command input queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean commandLevel(MQLONG &*Stufe*);
Stellt eine Kopie der Befehlsebene (**command level**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG commandLevel( );
Gibt die Befehlsebene (**command level**) ohne Angabe möglicher Fehler zurück.

MQLONG commandServerControl ( );
Gibt den Startwert für den Befehlsserver ohne Angabe möglicher Fehler zurück.

ImqBoolean commandServerControl (MQLONG & server);
Stellt eine Kopie des Startwerts für die Befehlsserversteuerung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean commit( );
Schreibt nicht festgeschriebene Änderungen fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean connect( );
Stellt eine Verbindung zum Warteschlangenmanager mit dem angegebenen ImqObject **-Namen**her. Der Standardwert ist der lokale Warteschlangenmanager. Wenn Sie eine Verbindung zu einem bestimmten Warteschlangenmanager herstellen wollen, müssen Sie vor der Verbindung die Methode **setName** von "ImqObject" verwenden. Wenn ein Kanalverweis (**channel reference**) vorhanden ist, wird dieser verwendet, um Informationen zur Kanaldefinition an MQCONNX in einer MQCD-Struktur zu übergeben. Der Kanaltyp in MQCD ist auf MQCHT_CLNTCONN gesetzt. **Kanalreferenzinformationen** , die nur für Clientverbindungen von Bedeutung sind, werden für Serververbindung ignoriert. Die Verbindungsoptionen (**connect options**) wirken sich auf das Verhalten dieser Methode aus. Diese Methode setzt bei erfolgreicher Ausführung den Verbindungsstatus (**connection status**) auf TRUE. Sie gibt den neuen Verbindungsstatus zurück.

Wenn es einen ersten Authentifizierungsdatensatz gibt, wird die Kette von Authentifizierungsdatensätzen verwendet, um digitale Zertifikate für sichere Clientkanäle zu authentifizieren.

Sie können mehrere ImqQueueManager-Objekte mit demselben Warteschlangenmanager verbinden. Alle Objekte verwenden dieselbe MQHCONN-Verbindungskennung und nutzen gemeinsam die UOW-Funktionen für die dem Thread zugeordnete Verbindung. Der erste "ImqQueueManager", der eine Verbindung herstellt, erhält die MQHCONN-Kennung. Der "ImqQueueManager", dessen Verbindung zuletzt getrennt wird, führt MQDISC aus.

Für ein Multithread-Programm wird empfohlen, dass für jeden Thread ein separates ImqQueueManager-Objekt verwendet wird.

ImqBinary connectionId ( ) const ;
Gibt die Verbindungs-ID zurück.

ImqBinary connectionTag ( ) const ;
Gibt das Verbindungstag (**connection tag**) zurück.

ImqBoolean setConnectionTag (const MQBYTE128 *Tag* = 0);
Legt das Verbindungstag (**connection tag**) fest. Wenn der Wert für *tag* null ist, wird das Verbindungstag (**connection tag**) gelöscht. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setConnectionTag (const ImqBinary & *Tag*);
Legt das Verbindungstag (**connection tag**) fest. Die Datenlänge (**data length**) von *tag* muss entweder null sein (um das Verbindungstag (**connection tag**) zu löschen) oder MQ_CONN_TAG_LENGTH aufweisen. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

MQLONG connectOptions( ) const ;
Gibt die Verbindungsoptionen (**connect options**) zurück.

void setConnectOptionen (const MQLONG *Optionen* = MQCNO);
Legt die Verbindungsoptionen (**connect options**) fest.

ImqBoolean connectionStatus( ) const ;
Gibt den Verbindungsstatus (**connection status**) zurück.

ImqString cryptographicHardware ( );
Gibt die Verschlüsselungshardware (**cryptographic hardware**) zurück.

ImqBoolean setCryptographicHardware (const char * *hardware* = 0);
Legt die Verschlüsselungshardware (**cryptographic hardware**) fest. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean deadLetterQueueName(ImqString & *Name*);
Stellt eine Kopie des Namens der Warteschlange für nicht zustellbare Nachrichten (**dead-letter queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString deadLetterQueueName( );
Gibt eine Kopie des Namens der Warteschlange für nicht zustellbare Nachrichten (**dead-letter queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean defaultTransmissionQueueName(ImqString & *Name*);
Stellt eine Kopie des Namens der Standardübertragungswarteschlange (**default transmission queue name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString defaultTransmissionQueueName( );
Gibt den Namen der Standardübertragungswarteschlange (**default transmission queue name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean disconnect( );
Trennt die Verbindung zum Warteschlangenmanager und setzt den Verbindungsstatus (**connection status**) auf FALSE. Schließt alle "ImqProcess"- und "ImqQueue"-Objekte, die diesem Objekt zugeordnet sind, und unterbricht vor dem Abbau der Verbindung deren Verbindungsverweis (**connection reference**). Wenn mehrere "ImqQueueManager"-Objekte mit demselben Warteschlangenmanager verbunden sind, führt nur derjenige, dessen Verbindung als letzte getrennt wird, einen physischen Verbindungsabbau aus; die anderen führen einen logischen Verbindungsabbau aus. Nicht festgeschriebene Änderungen werden nur bei physischem Verbindungsabbau festgeschrieben.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben. Wird sie bei nicht vorhandener Verbindung aufgerufen, wird der Rückkehrcode ebenfalls auf TRUE gesetzt.

ImqBoolean distributionLists(MQLONG & *-Unterstützung*);
Stellt eine Kopie des Werts für die Verteilerlisten (**distribution lists**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG distributionLists( );
Gibt den Wert für die Verteilerlisten (**distribution lists**) ohne Angabe möglicher Fehler zurück.

ImqBoolean dnsGroup (ImqString & group);
Stellt eine Kopie des DNS-Gruppennamens bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString dnsGroup ( );
Gibt den DNS-Gruppennamen ohne Angabe möglicher Fehler zurück.

ImqBoolean dnsWlm (MQLONG & wlm);
Stellt eine Kopie des Werts für das DNS-Auslastungsmanagement bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG dnsWlm ( );
Gibt den Wert für das DNS-Auslastungsmanagement ohne Angabe möglicher Fehler zurück.

ImqAuthenticationRecord * firstAuthenticationRecord ( ) const ;
Gibt den ersten Authentifizierungsdatensatz (**first authentication record**) zurück.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * *air* = 0);
Legt den ersten Authentifizierungsdatensatz (**first authentication record**) fest.

ImqObject * firstManagedObject( ) const ;
Gibt das erste verwaltete Objekt (**first managed object**) zurück.

ImqBoolean inhibitEvent(MQLONG & *ereignis*);
Stellt eine Kopie des Aktivierungsstatus des Sperrereignisses (**inhibit event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG inhibitEvent( );
Gibt den Aktivierungsstatus des Sperrereignisses (**inhibit event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean ipAddressVersion (MQLONG & version);
Stellt eine Kopie des Werts für die Version der IP-Adresse bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG ipAddressVersion ( );
Gibt den Wert für die Version der IP-Adresse ohne Angabe möglicher Fehler zurück.

ImqBoolean keepAlive (MQLONG & keepalive);
Gibt eine Kopie des Keepalive-Werts zurück. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG keepAlive ( );
Gibt den Keepalive-Wert ohne Angabe möglicher Fehler zurück.

ImqString keyRepository ( );
Gibt das Schlüsselrepository (**key repository**) zurück.

ImqBoolean setKeyRepository (const char * *Repository* = 0);
Legt das Schlüsselrepository (**key repository**) fest. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean listenerTimer (MQLONG & timer);
Stellt eine Kopie des Werts für den Zeitgeber des Empfangsprogramms bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG listenerTimer ( );
Gibt den Wert für den Zeitgeber des Empfangsprogramms ohne Angabe möglicher Fehler zurück.

ImqBoolean localEvent(MQLONG & *Ereignis*);
Stellt eine Kopie des Aktivierungsstatus des lokalen Ereignisses (**local event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG localEvent( );
Gibt den Aktivierungsstatus des lokalen Ereignisses (**local event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean loggerEvent (MQLONG & count);
Stellt eine Kopie des Werts für das Protokollierungsereignis bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG loggerEvent ( );
Gibt den Wert für das Protokollierungsereignis ohne Angabe möglicher Fehler zurück.

ImqBoolean luGroupName (ImqString & name);.
Stellt eine Kopie des LU-Gruppennamens bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString luGroupName ( );
Gibt den LU-Gruppennamen ohne Angabe möglicher Fehler zurück.

ImqBoolean lu62ARMSuffix (ImqString & suffix);
Stellt eine Kopie des LU62-ARM-Suffixes bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString lu62ARMSuffix ( );
Gibt das LU62-ARM-Suffix ohne Angabe möglicher Fehler zurück.

ImqBoolean luName (ImqString & name);
Stellt eine Kopie des LU-Namens bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString luName ( );
Gibt den LU-Namen ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumActiveChannels (MQLONG & channels);
Stellt eine Kopie des Werts für die maximale Anzahl aktiver Kanäle bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumActiveChannels ( );
Gibt den Wert für die maximale Anzahl aktiver Kanäle ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumCurrentChannels (MQLONG & channels);
Stellt eine Kopie des Werts für die maximale Anzahl aktueller Kanäle bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumCurrentChannels ( );
Gibt den Wert für die maximale Anzahl aktueller Kanäle ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumHandles(MQLONG & ***Anzahl*** **);**
Stellt eine Kopie der maximalen Anzahl Kennungen (**maximum handles**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumHandles( );
Gibt die maximale Anzahl Kennungen (**maximum handles**) ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumLu62Channels (MQLONG & channels);
Stellt eine Kopie des Werts für die maximale Anzahl LU62-Kanäle bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumLu62Channels ( );.
Gibt den Wert für die maximale Anzahl LU62-Kanäle ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumMessageLength (MQLONG & ***länge*** **);**
Stellt eine Kopie der maximalen Nachrichtenlänge (**maximum message length**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumMessageLength( );
Gibt die maximale Nachrichtenlänge (**maximum message**) ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumPriority(MQLONG & ***priorität*** **);**
Stellt eine Kopie der maximalen Priorität (**maximum priority**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumPriority( );
Gibt eine Kopie der maximalen Priorität (**maximum priority**) ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumTcpChannels (MQLONG & channels);
Stellt eine Kopie des Werts für die maximale Anzahl TCP-Kanäle bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumTcpChannels ( );
Gibt den Wert für die maximale Anzahl TCP-Kanäle ohne Angabe möglicher Fehler zurück.

ImqBoolean maximumUncommittedNachrichten (MQLONG & ***number*** **);**
Stellt eine Kopie der maximalen Anzahl nicht festgeschriebener Nachrichten (**maximum uncommitted messages**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG maximumUncommittedMessages( );
Gibt die maximale Anzahl nicht festgeschriebener Nachrichten (**maximum uncommitted messages**) ohne Angabe möglicher Fehler zurück.

ImqBoolean mqiAccounting (MQLONG & statint);
Stellt eine Kopie des Werts für die MQI-Abrechnung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG mqiAccounting ( );
Gibt den Wert für die MQI-Abrechnung ohne Angabe möglicher Fehler zurück.

ImqBoolean mqiStatistics (MQLONG & statmqi);
Stellt eine Kopie des Werts für die MQI-Statistik bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG mqiStatistics ( );
Gibt den Wert für die MQI-Statistik ohne Angabe möglicher Fehler zurück.

ImqBoolean outboundPortMax (MQLONG & max);
Stellt eine Kopie des maximalen Werts für den Port für abgehende Daten bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG outboundPortMax ( );
Gibt den maximalen Wert für den Port für abgehende Daten ohne Angabe möglicher Fehler zurück.

ImqBoolean outboundPortMin (MQLONG & min);
Stellt eine Kopie des Mindestwerts für den Port für abgehende Daten bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG outboundPortMin ( );
Gibt den Mindestwert für den Port für abgehende Daten ohne Angabe möglicher Fehler zurück.

ImqBinary password ( ) const;
Gibt das für Clientverbindungen verwendete Kennwort zurück.

ImqBoolean setPassword (const ImqString & password);
Legt das für Clientverbindungen verwendete Kennwort fest.

ImqBoolean setPassword (const char * = 0 password);
Legt das für Clientverbindungen verwendete Kennwort fest.

ImqBoolean setPassword (const ImqBinary & password);
Legt das für Clientverbindungen verwendete Kennwort fest.

ImqBoolean performanceEvent(MQLONG & *Ereignis*);
Stellt eine Kopie des Aktivierungsstatus des Leistungsereignisses (**performance event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG performanceEvent( );
Gibt den Aktivierungsstatus des Leistungsereignisses (**performance event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean -Plattform (MQLONG & *platform*)
Stellt eine Kopie der Plattform (**platform**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG platform( );
Gibt die Plattform (**platform**) ohne Angabe möglicher Fehler zurück.

ImqBoolean queueAccounting (MQLONG & acctq);
Stellt eine Kopie des Werts für die Warteschlangenabrechnung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueAccounting ( );
Gibt den Wert für die Warteschlangenabrechnung ohne Angabe möglicher Fehler zurück.

ImqBoolean queueMonitoring (MQLONG & monq);
Stellt eine Kopie des Werts für die Warteschlangenüberwachung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueMonitoring ( );
Gibt den Wert für die Warteschlangenüberwachung ohne Angabe möglicher Fehler zurück.

ImqBoolean queueStatistics (MQLONG & statq);
Stellt eine Kopie des Werts für die Warteschlangenstatistik bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG queueStatistics ( );
Gibt den Wert für die Warteschlangenstatistik ohne Angabe möglicher Fehler zurück.

ImqBoolean receiveTimeout (MQLONG & timeout);
Stellt eine Kopie des Werts für das Empfangszeitlimit bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG receiveTimeout ( );
Gibt den Wert für das Empfangszeitlimit ohne Angabe möglicher Fehler zurück.

ImqBoolean receiveTimeoutMin (MQLONG & min);
Stellt eine Kopie des Mindestwerts für das Empfangszeitlimit bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG receiveTimeoutMin ( );
Gibt den Mindestwert für das Empfangszeitlimit ohne Angabe möglicher Fehler zurück.

ImqBoolean receiveTimeoutType (MQLONG & type);
Stellt eine Kopie des Empfangszeitlimittyps bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG receiveTimeoutType ( );
Gibt den Empfangszeitlimittyp ohne Angabe möglicher Fehler zurück.

ImqBoolean remoteEvent(MQLONG & ***Ereignis*** **);**
Stellt eine Kopie des Aktivierungsstatus des fernen Ereignisses (**remote event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG remoteEvent( );
Gibt den Aktivierungsstatus des fernen Ereignisses (**remote event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean repositoryName(ImqString & ***Name*** **);**
Stellt eine Kopie des **Repository-Namens**bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString repositoryName( );
Gibt den **Repository-Namen** ohne Angabe möglicher Fehler zurück.

ImqBoolean repositoryNamelistName (ImqString & ***Name*** **)**
Stellt eine Kopie des Namens der Repository-Namensliste (**repository namelist name**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString repositoryNamelistName( );
Gibt eine Kopie des Namens der Repository-Namensliste (**repository namelist name**) ohne Angabe möglicher Fehler zurück.

ImqBoolean sharedQueueQueueManagerName (MQLONG & name);
Stellt eine Kopie des Werts für den Warteschlangenmanagernamen der gemeinsam genutzten Warteschlange bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG sharedQueueQueueManagerName ( );
Gibt den Wert für den Warteschlangenmanagernamen der gemeinsam genutzten Warteschlange ohne Angabe möglicher Fehler zurück.

ImqBoolean sslEvent (MQLONG & event);
Stellt eine Kopie des Werts für das SSL-Ereignis bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG sslEvent ( );
Gibt den Wert für das SSL-Ereignis ohne Angabe möglicher Fehler zurück.

ImqBoolean sslFips (MQLONG & sslfips);
Stellt eine Kopie des SSL-FIPS-Werts bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG sslFips ( );
Gibt den SSL-FIPS-Wert ohne Angabe möglicher Fehler zurück.

ImqBoolean sslKeyResetCount (MQLONG & count);
Stellt eine Kopie des Werts für die Anzahl der Rücksetzungen für SSL-Schlüssel bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG sslKeyResetCount ( );
Gibt den Wert für die Anzahl der Rücksetzungen für SSL-Schlüssel ohne Angabe möglicher Fehler zurück.

ImqBoolean startStopEreignis (MQLONG & *Ereignis*);
Stellt eine Kopie des Aktivierungsstatus des Start-Stopp-Ereignisses (**start-stop event**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG startStopEvent( );
Gibt den Aktivierungsstatus des Start-Stopp-Ereignisses (**start-stop event**) ohne Angabe möglicher Fehler zurück.

ImqBoolean statisticsInterval (MQLONG & statint);
Stellt eine Kopie des Werts für das Statistikintervall bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG statisticsInterval ( );
Gibt den Wert für das Statistikintervall ohne Angabe möglicher Fehler zurück.

ImqBoolean syncPointVerfügbarkeit (MQLONG & *sync*);
Stellt eine Kopie des Werts für die Synchronisationspunktverfügbarkeit (**syncpoint availability**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG syncPointAvailability( );
Gibt eine Kopie des Werts für die Synchronisationspunktverfügbarkeit (**syncpoint availability**) ohne Angabe möglicher Fehler zurück.

ImqBoolean tcpName (ImqString & name);
Stellt eine Kopie des TCP-Systemnamens bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqString tcpName ( );
Gibt den TCP-Systemnamen ohne Angabe möglicher Fehler zurück.

ImqBoolean tcpStackType (MQLONG & type);
Stellt eine Kopie des TCP-Stapeltyps bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG tcpStackType ( );
Gibt den TCP-Stapeltyp ohne Angabe möglicher Fehler zurück.

ImqBoolean traceRouteRecording (MQLONG & routerec);
Stellt eine Kopie des Werts für die Traceroute-Aufzeichnung bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG traceRouteRecording ( );
Gibt den Wert für die Traceroute-Aufzeichnung ohne Angabe möglicher Fehler zurück.

ImqBoolean triggerInterval(MQLONG & *intervall*);
Stellt eine Kopie des Auslöseintervalls (**trigger interval**) bereit. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

MQLONG triggerInterval( );
Gibt das Auslöseintervall (**trigger interval**) ohne Angabe möglicher Fehler zurück.

ImqBinary userId ( ) const;
Gibt die für Clientverbindungen verwendete Benutzer-ID zurück.

ImqBoolean setUserId (const ImqString & id);
Legt die für Clientverbindungen verwendete Benutzer-ID fest.

ImqBoolean setUserId (const char * = 0 id);
Legt die für Clientverbindungen verwendete Benutzer-ID fest.

ImqBoolean setUserId (const ImqBinary & id);
Legt die für Clientverbindungen verwendete Benutzer-ID fest.

Objektmethoden (geschützt)

void setFirstManagedObject(Const ImqObject * *Objekt* = 0);
Legt das erste verwaltete Objekt (**first managed object**) fest.

Objektdaten (geschützt)

MQHCONN *ohconn*
Die WebSphere MQ-Verbindungskennung (nur aussagekräftig, wenn der Verbindungsstatus (**connection status**) auf TRUE gesetzt ist).

Ursachencodes

- MQRC_ATTRIBUTE_LOCKED
- MQRC_ENVIRONMENT_ERROR
- MQRC_FUNCTION_NOT_SUPPORTED
- MQRC_REFERENCE_ERROR
- (Ursachencodes für MQBACK)
- (Ursachencodes für MQBEGIN)
- (Ursachencodes für MQCMIT)
- (Ursachencodes für MQCONNX)
- (Ursachencodes für MQDISC)
- (Ursachencodes für MQCONN)

C++-Klasse "ImqReferenceHeader"

Diese Klasse bindet Funktionen der MQRMH-Datenstruktur ein.

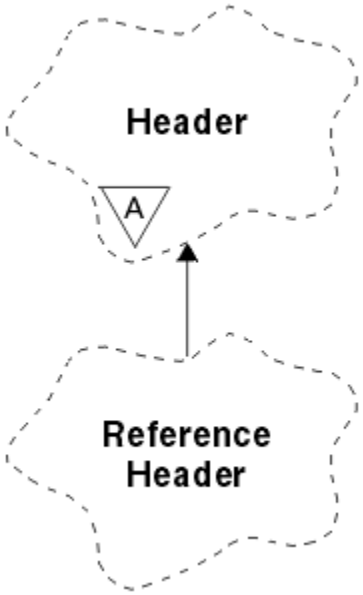


Abbildung 66. ImqReferenceHeader, Klasse

Diese Klasse bezieht sich auf die im Abschnitt „Querverweise für ImqReferenceHeader” auf Seite 1358 aufgelisteten MQI-Aufrufe.

Objektattribute

destination environment
Umgebung für die Zieladresse. Der Anfangswert ist eine Nullzeichenfolge.

destination name
Name des Datenziels. Der Anfangswert ist eine Nullzeichenfolge.

instance id
Instanz-ID. Ein binärer Wert (MQBYTE24) mit der Länge MQ_INSTANCE_ID_LENGTH. Der Anfangswert ist MQOII_NONE.

logical length
Logische bzw. vorgesehene Länge von Nachrichtendaten, die nach diesem Header folgen. Der Anfangswert ist null.

logical offset
Logische relative Adresse für die nachfolgenden Nachrichtendaten, die an der letzten Zieladresse im Kontext der gesamten Daten interpretiert werden soll. Der Anfangswert ist null.

logical offset 2
Höchstwertige Erweiterung für die logische relative Adresse (**logical offset**). Der Anfangswert ist null.

reference type
Verweistyp. Der Anfangswert ist eine Nullzeichenfolge.

source environment
Umgebung für die Quelle. Der Anfangswert ist eine Nullzeichenfolge.

source name
Name der Datenquelle. Der Anfangswert ist eine Nullzeichenfolge.

Konstruktoren

ImqReferenceHeader( );
Der Standardkonstruktor.

ImqReferenceHeader (const ImqReferenceHeader & *Header*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Fügt am Anfang eine MQRMH-Datenstruktur in den Nachrichtenpuffer ein, verschiebt vorhandene Nachrichtendaten weiter und setzt das *msg* **-Format** auf MQFMT_REF_MSG_HEADER.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqHeader" im Abschnitt „C++-Klasse "ImqHeader"” auf Seite 1388.

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Liest eine MQRMH-Datenstruktur aus dem Nachrichtenpuffer.

Um erfolgreich zu sein, muss das ImqMessage **-Format** MQFMT_REF_MSG_HEADER sein.

Weitere Informationen finden Sie in der Beschreibung der Methoden der Klasse "ImqHeader" im Abschnitt „C++-Klasse "ImqHeader"” auf Seite 1388.

Objektmethoden (öffentlich)

void operator = (const ImqReferenceHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqString destinationEnvironment( ) const ;
Gibt eine Kopie der Zielumgebung (**destination environment**) zurück.

void setDestinationEnvironment(const char * *environment* = 0);
Legt die Zielumgebung (**destination environment**) fest.

ImqString destinationName( ) const ;
Gibt eine Kopie des Bestimmungsnamens (**destination name**) zurück.

void setDestinationName(const char * *name* = 0);
Legt den Zielnamen (**destination name**) fest.

ImqBinary instanceId( ) const ;
Gibt eine Kopie der Instanz-ID (**instance id**) zurück.

ImqBoolean setInstanceId(const ImqBinary & *id*);
Legt die **Instanz-ID**fest. Die Datenlänge (**data length**) von *token* muss entweder 0 oder MQ_OBJECT_INSTANCE_ID_LENGTH sein. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

void setInstanceId(const MQBYTE24 *id* = 0);
Legt die **Instanz-ID**fest. Der Wert für *id* kann null sein, was der Angabe von MQOII_NONE entspricht. Wenn der Wert für *id* ungleich null ist, muss er MQ_OBJECT_INSTANCE_LENGTH Bytes an binären Daten aufweisen. Werden vordefinierte Werte wie beispielsweise MQOII_NONE verwendet, müssen Sie möglicherweise eine Umsetzung durchführen, um eine Übereinstimmung der Signaturen sicherzustellen, z. B. (MQBYTE *)MQOII_NONE.

MQLONG logicalLength( ) const ;
Gibt die logische Länge (**logical length**) zurück.

void setLogicalLength(const MQLONG *length*);
Legt die logische Länge (**logical length**) fest.

MQLONG logicalOffset( ) const ;
Gibt die logische relative Adresse (**logical offset**) zurück.

void setLogicalOffset(const MQLONG *offset*);
Legt die logische relative Adresse (**logical offset**) fest.

MQLONG logicalOffset2( ) const ;
Gibt die 2. logische relative Adresse (**logical offset 2**).

void setLogicalOffset2(const MQLONG *offset*);
Legt die 2. logische relative Adresse (**logical offset 2**) fest.

ImqString referenceType( ) const ;
Gibt eine Kopie des Verweistyps (**reference type**) zurück.

void setReferenceType(const char * *name* = 0);
Legt den Verweistyp (**reference type**) fest.

ImqString sourceEnvironment( ) const ;
Gibt eine Kopie der Quellenumgebung (**source environment**) zurück.

void setSourceEnvironment(const char * *environment* = 0);
Legt die Quellenumgebung (**source environment**) fest.

ImqString sourceName( ) const ;
Gibt eine Kopie des Quellennamens (**source name**) zurück.

void setSourceName(const char * *name* = 0);
Legt den Quellennamen (**source name**) fest.

Objektdaten (geschützt)

MQRMH *omqrmh*
Die MQRMH-Datenstruktur.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INSUFFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR

C++-Klasse ImqString

Diese Klasse stellt die Zeichenfolgenspeicherung und Bearbeitung für auf NULL endende Zeichenfolgen bereit.

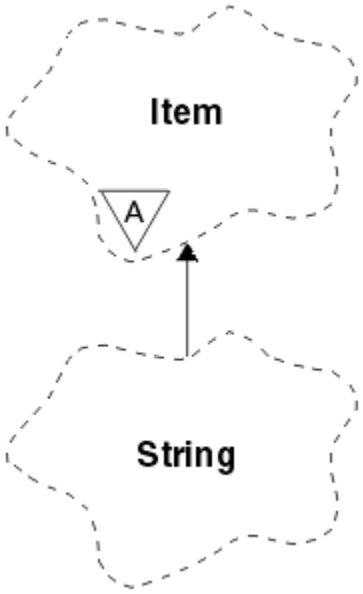


Abbildung 67. ImqString-Klasse

Sie können ImqString anstelle von **char *** in den meisten Situationen verwenden, in denen ein Parameter **char *** erfordert.

- „Objektattribute” auf Seite 1448
- „Konstruktoren” auf Seite 1449
- „Klassenmethoden (öffentlich)” auf Seite 1449
- „Methoden für überlastete ImqItem-Klassen” auf Seite 1449
- „Objektmethoden (öffentlich)” auf Seite 1450
- „Objektmethoden (geschützt)” auf Seite 1453
- „Ursachencodes” auf Seite 1453

Objektattribute

characters
Zeichen im Speicher (**storage**), denen eine abschließende Null vorausgeht.

Länge
Anzahl der Bytes in den Zeichen (**characters**). Wenn kein Wert für den Speicher (**storage**) angegeben ist, beträgt der Wert für die Länge (**length**) null. Der Anfangswert ist null.

storage
Ein flüchtiger Bytebereich von beliebiger Größe. Im Speicher (**storage**) muss nach den Zeichen (**characters**) immer eine abschließende Null vorhanden sein, damit das Ende der Zeichen (**characters**) erkannt wird. Mit Methoden wird sichergestellt, dass diese Situation beibehalten wird. Wenn Sie die Bytes jedoch direkt im Bereich festlegen, stellen Sie vor der Bearbeitung sicher, dass eine abschließende Null vorhanden ist. Anfangs liegt kein **storage**-Attribut vor.

Konstruktoren

ImqString( );
Der Standardkonstruktor.

ImqString(const ImqString & *Zeichenfolge*);
Der Kopierkonstruktor.

ImqString(const char *c*);
Die Zeichen (**characters**) umfassen *c*.

ImqString(const char * *text*);
Die Zeichen (**characters**) werden aus dem Text (*text*) kopiert.

ImqString(const void * *buffer*, const size_t *length*);
Kopiert die als *length* definierte Anzahl von Bytes ab *buffer* (Puffer) und weist diese den Zeichen (**characters**) zu. Die Ersetzung erfolgt für alle kopierten Nullzeichen. Das Substitutionszeichen ist ein Punkt (.). Es werden keine anderen nicht druckbaren oder nicht anzeigbaren Zeichen, die kopiert werden, berücksichtigt.

Klassenmethoden (öffentlich)

static ImqBoolean copy (char * *Zielpuffer*, const size_t *Länge*, const char * *Quellenpuffer*, const char *pad* = 0);
Kopiert Bytes bis zur angegebenen Länge (*length*) aus dem Quellenpuffer (*source-buffer*) in den Zielpuffer (*destination-buffer*). Wenn die Anzahl der Zeichen im *Quellenpuffer* nicht ausreicht, wird der restliche Platz im *Zielpuffer* mit *Füllzeichen* aufgefüllt. Der *Quellenpuffer* darf den Wert null aufweisen. Der *Zielpuffer* darf den Wert null aufweisen, wenn die *Länge* ebenfalls null beträgt. Alle Fehlercodes gehen verloren. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

static ImqBoolean copy (char * *Zielpuffer*, const size_t *Länge*, const char * *Quellenpuffer*, ImqError & *Fehlerobjekt*, const char *pad* = 0);
Kopiert Bytes bis zur angegebenen Länge (*length*) aus dem Quellenpuffer (*source-buffer*) in den Zielpuffer (*destination-buffer*). Wenn die Anzahl der Zeichen im *Quellenpuffer* nicht ausreicht, wird der restliche Platz im *Zielpuffer* mit *Füllzeichen* aufgefüllt. Der *Quellenpuffer* darf den Wert null aufweisen. Der *Zielpuffer* darf den Wert null aufweisen, wenn die *Länge* ebenfalls null beträgt. Alle Fehlercodes werden in *error-object* festgelegt. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Kopiert die Zeichen (**characters**) in den Nachrichtenpuffer und ersetzt dabei den gesamten bereits vorhandenen Inhalt. Setzt das *msg*-**Format** auf MQFMT_STRING.

Weitere Informationen finden Sie in der Beschreibung der übergeordneten Klassenmethode.

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Legt die Zeichen (**characters**) durch die Übertragung der verbleibenden Daten aus dem Nachrichtenpuffer fest. Dabei werden die bereits vorhandenen **Zeichen** ersetzt.

Damit dieser Vorgang erfolgreich ist, muss als Codierung (**encoding**) des *msg*-Objekts MQENC_NATIVE verwendet werden. Abrufen von Nachrichten mit MQGMO_CONVERT nach MQENC_NATIVE.

Damit dieser Vorgang erfolgreich ist, muss das ImqMessage-**Format** MQFMT_STRING lauten.

Weitere Informationen finden Sie in der Beschreibung der übergeordneten Klassenmethode.

Objektmethoden (öffentlich)

char & operator [] (const size_t *offset*) Const;
Verweist auf das Zeichen im Absatz *offset* im **Speicher**. Stellen Sie sicher, dass das entsprechende Byte vorhanden und adressierbar ist.

Operator ImqString () (const size_t *Offset*, const size_t *Länge* = 1) Const;
Gibt eine Unterzeichenfolge aus, indem die Bytes aus den **Zeichen** ab *offset* kopiert werden. Wenn die Länge (*length*) null beträgt, wird der Rest der **Zeichen** zurückgegeben. Wenn die Kombination aus relativer Position (*offset*) und Länge (*length*) keine Referenz innerhalb der **Zeichen** produziert, wird ein leerer ImqString-Wert zurückgegeben.

void operator = (const ImqString & *Zeichenfolge*);
Kopiert Instanzdaten aus *string* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqString operator + (const char *c*) const ;
Gibt das Ergebnis beim Anhängen von *c* an die **Zeichen** zurück.

ImqString operator + (const char * *text*) Const;
Gibt das Ergebnis beim Anhängen von *text* an die **Zeichen** zurück. Dieser Vorgang kann auch umgekehrt werden. Beispiel:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Anmerkung: Auch wenn die meisten Compiler **strOne + "string two";** akzeptieren, erfordert Microsoft Visual C++ die Eingabe von **strOne + (char *)"string two" ;**

ImqString operator + (const ImqString & *string1*) Const;
Gibt das Ergebnis beim Anhängen von *string1* an die **Zeichen** zurück.

ImqString operator + (const double *zahl*) Const;
Gibt das Ergebnis beim Anhängen von *number* an die **Zeichen** nach der Konvertierung von Text zurück.

ImqString operator + (const long *zahl*) Const;
Gibt das Ergebnis beim Anhängen von *number* an die **Zeichen** nach der Konvertierung von Text zurück.

void operator += (const char *c*);
Hängt *c* an die **Zeichen** an.

void operator + = (const char * *Text*);
Hängt *text* an die **Zeichen** an.

void operator + = (const ImqString & *Zeichenfolge*);
Hängt *string* an die **Zeichen** an.

void operator + = (const double *Zahl*);
Hängt *number* nach der Konvertierung in Text an die **Zeichen** an.

void operator + = (const long *Zahl*);
Hängt *number* nach der Konvertierung in Text an die **Zeichen** an.

operator char * ( ) const ;
Gibt die Adresse des ersten Bytes im **Speicher** zurück. Dieser Wert darf nicht null sein und ist flüchtig. Verwenden Sie diese Methode nur für schreibgeschützte Zwecke.

ImqBoolean operator < (const ImqString & *zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Das Ergebnis ist TRUE bei kleiner als und FALSE bei größer-gleich.

ImqBoolean -Operator > (const ImqString & *Zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Das Ergebnis ist TRUE bei größer als und FALSE bei kleiner-gleich.

ImqBoolean operator < = (const ImqString & *zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Das Ergebnis ist TRUE bei kleiner-gleich und FALSE bei größer als.

ImqBoolean Operator > = (const ImqString & *Zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Das Ergebnis ist TRUE bei größer-gleich und FALSE bei kleiner als.

Operator ImqBoolean == (const ImqString & *Zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Es wird entweder TRUE oder FALSE zurückgegeben.

Operator ImqBoolean ! = (const ImqString & *Zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Dabei wird die Vergleichsmethode (**compare**) verwendet. Es wird entweder TRUE oder FALSE zurückgegeben.

short compare (const ImqString & *Zeichenfolge*) Const;
Vergleicht die **Zeichen** mit denen unter *string*. Das Ergebnis ist null, wenn die **Zeichen** gleich sind, negativ bei kleiner als und positiv bei größer als. Beim Vergleich muss die Groß-/Kleinschreibung beachtet werden. Eine ImqString-Klasse mit dem Wert null gilt als kleiner als eine ImqString-Klasse ungleich null.

ImqBoolean copyOut(char * *Puffer*, const size_t *Länge*, const char *pad* = 0);
Kopiert Bytes bis zur angegebenen Länge (*length*) aus den **Zeichen** in den *Puffer*. Wenn die Anzahl der **Zeichen** nicht ausreicht, wird der restliche Platz im *Puffer* mit *Füllzeichen* aufgefüllt. Der *Puffer* darf den Wert null aufweisen, wenn die *Länge* ebenfalls null beträgt. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

size_t copyOut(lang & *Anzahl*) Const;
Legt *number* anhand der **Zeichen** nach der Konvertierung aus Text fest und gibt die Anzahl der an der Konvertierung beteiligten Zeichen zurück. Wenn dieser Wert null lautet, wurde keine Konvertierung durchgeführt, und *number* wurde nicht festgelegt. Eine konvertierbare Zeichenfolge muss mit den folgenden Werten beginnen:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & *token*, const char *c* = '') Const;
Wenn unter **characters** mindestens ein Zeichen enthalten ist, das sich von *c* unterscheidet, wird ein Token als erste zusammenhängende Folge solcher Zeichen angegeben. In diesem Fall wird *token* auf diese Folge gesetzt, und der zurückgegebene Wert ist die Summe der Anzahl führender Zeichen *c* und der Anzahl der Bytes in der Folge. Andernfalls wird null zurückgegeben und *token* nicht festgelegt.

size_t cutOut(long & *Anzahl*);
Legt *number* wie für die Kopiermethode (**copy**) fest, entfernt jedoch die vom Rückgabewert angegebene Anzahl Bytes aus **characters** (Zeichen). Beispiel: Die im folgenden Beispiel angezeigte Zeichenfolge kann durch dreimaliges Ausführen des Befehls **cutOut**(*number*) in drei Zahlen aufgeteilt werden:

```
strNumbers = "-1 0     +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & *Token*, const char *c* = ''
Legt *token* wie für die **copyOut**-Methode fest und entfernt aus **characters** die *strToken*-Zeichen sowie alle Zeichen *c*, die den *token*-Zeichen vorangestellt sind. Wenn *c* kein Leerzeichen ist, werden die Zeichen *c* entfernt, die direkt auf die *token*-Zeichen folgen. Gibt die Anzahl der entfernten Zeichen

zurück. Beispiel: Die im folgenden Beispiel angezeigte Zeichenfolge kann durch dreimaliges Ausführen des Befehls **cutOut**(*token*) in drei Token aufgeteilt werden:

```
strText = "  Program Version 1.1  "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == "  "
```

Im folgenden Beispiel wird dargestellt, wie Sie einen DOS-Pfadnamen syntaktisch analysieren:

```
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 & *Zeichenfolge*);
Sucht nach einer exakten Übereinstimmung für *string* an einer beliebigen Stelle unter **characters**. Wenn keine Übereinstimmung gefunden wird, wird FALSE zurückgegeben. Andernfalls wird TRUE zurückgegeben. Wenn *string* den Wert null aufweist, wird TRUE zurückgegeben.

ImqBoolean find (const ImqString & *Zeichenfolge*, size_t & *offset*);
Sucht nach einer exakten Übereinstimmung für *string* an einer beliebigen Stelle unter **characters** (Zeichen) ab der relativen Position (*offset*). Wenn *string* den Wert null aufweist, wird TRUE zurückgegeben, *offset* jedoch nicht aktualisiert. Wenn keine Übereinstimmung gefunden wird, wird FALSE zurückgegeben (der Wert für *offset* hat sich möglicherweise erhöht). Wenn eine Übereinstimmung gefunden wird, wird TRUE zurückgegeben und *offset* auf die relative Position von *string* innerhalb von **characters** aktualisiert.

size_t length( ) const ;
Gibt die Länge (**length**) zurück.

ImqBoolean pasteIn(const double *Anzahl*, const char * *Format* = "%f");
Hängt *number* nach der Konvertierung in Text an die **Zeichen** an. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Die Spezifikation *format* wird zur Formatierung der Gleitkommakonvertierung verwendet. Wenn dieser Wert angegeben wird, muss er für die Verwendung mit **printf** und Gleitkommazahlen geeignet sein, z.B. für **%.3f**.

ImqBoolean pasteIn(const long *zahl*);
Hängt *number* nach der Konvertierung in Text an die **Zeichen** an. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

ImqBoolean pasteIn(const void * *buffer*, const size_t *länge*);
Hängt Bytes mit einer bestimmten *Länge* aus dem *Puffer* an die **Zeichen** an und fügt am Ende eine abschließende Null hinzu. Die Ersetzung erfolgt für alle kopierten Nullzeichen. Das Substitutionszeichen ist ein Punkt (.). Es werden keine anderen nicht druckbaren oder nicht anzeigbaren Zeichen, die kopiert werden, berücksichtigt. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean set(const char * *buffer*, const size_t *length*);
Legt die **Zeichen** aus einem Zeichenfeld mit fester Länge fest, die eine Null enthalten können. Hängt bei Bedarf eine Null an die Zeichen aus dem Feld mit fester Länge an. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqBoolean setStorage(const size_t *länge*);
Ordnet den Speicher (**storage**) zu (bzw. erneut zu). Behält alle ursprünglichen **Zeichen** einschließlich der abschließenden Null bei, wenn dafür ausreichend Platz zur Verfügung steht. Es wird jedoch kein zusätzlicher Speicher initialisiert.

Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

size_t storage( ) const ;
Gibt die Anzahl der Bytes im **Speicher** zurück.

size_t stripLeading(const char *c* = ' ');
Entfernt führende Zeichen *c* aus den **Zeichen** und gibt die Anzahl der entfernten Zeichen zurück.

size_t stripTrailing(const char *c* = ' ');
Entfernt abschließende Zeichen *c* aus den **Zeichen** und gibt die Anzahl der entfernten Zeichen zurück.

ImqString upperCase( ) const ;
Gibt eine großgeschriebene Kopie der **Zeichen** zurück.

Objektmethoden (geschützt)

ImqBoolean assign(const ImqString & *Zeichenfolge*);
Ist äquivalent zur funktional entsprechenden Methode **operator =**, nur nicht virtuell. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

Ursachencodes

- MQRC_DATA_TRUNCATED
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT

C++-Klasse ImqTrigger

Diese Klasse bindet die MQTM-Datenstruktur (Auslösenachricht) ein.

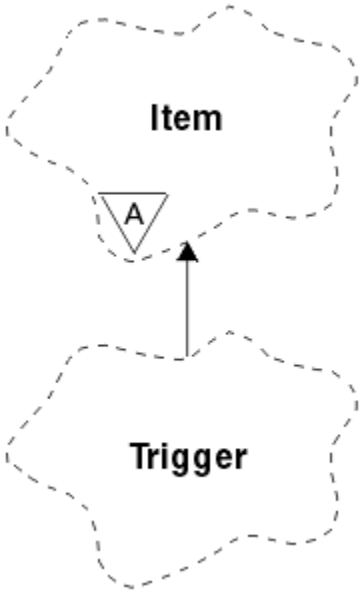


Abbildung 68. ImqTrigger-Klasse

Objekte dieser Klasse werden in der Regel von einem Auslösemonitorprogramm verwendet. Die Aufgabe eines Auslösemonitorprogramms besteht darin, auf diese speziellen Nachrichten zu warten und dann

entsprechend zu reagieren, um sicherzustellen, dass andere WebSphere MQ-Anwendungen gestartet werden, wenn Nachrichten auf diese Anwendungen warten.

Ein Verwendungsbeispiel finden Sie im IMQSTRG-Beispielprogramm.

Objektattribute

application id
ID der Anwendung, von der die Nachricht gesendet wurde. Der Anfangswert ist eine Nullzeichenfolge.

application type
Typ der Anwendung, von der die Nachricht gesendet wurde. Der Anfangswert ist null. Folgende zusätzliche Werte sind möglich:

- 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

environment data
Umgebungsdaten für den Prozess. Der Anfangswert ist eine Nullzeichenfolge.

process name
Name des Prozesses Der Anfangswert ist eine Nullzeichenfolge.

queue name
Der Name der Warteschlange, die gestartet werden soll. Der Anfangswert ist eine Nullzeichenfolge.

trigger data
Auslöserdaten für den Prozess. Der Anfangswert ist eine Nullzeichenfolge.

user data
Benutzerdaten für den Prozess. Der Anfangswert ist eine Nullzeichenfolge.

Konstruktoren

ImqTrigger( );
Der Standardkonstruktor.

ImqTrigger(const ImqTrigger & *Auslöser*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtual ImqBoolean copyOut(ImqMessage & *Nachricht*);
Schreibt eine MQTM-Datenstruktur in den Nachrichtenpuffer und ersetzt dabei den gesamten bereits vorhandenen Inhalt. Setzt das *msg*-**Format** auf MQFMT_TRIGGER.

Weitere Details finden Sie in der Beschreibung der Methoden der Klasse ImqItem unter „C++-Klasse "ImqItem"" auf Seite 1392.

virtuellen ImqBoolean pasteIn(ImqMessage & *msg*);
Liest eine MQTM-Datenstruktur aus dem Nachrichtenpuffer.

Damit dieser Vorgang erfolgreich ist, muss das ImqMessage-**Format** MQFMT_TRIGGER lauten.

Weitere Details finden Sie in der Beschreibung der Methoden der Klasse ImqItem unter „C++-Klasse "ImqItem"" auf Seite 1392.

Objektmethoden (öffentlich)

void operator = (const ImqTrigger & *trigger*);
Kopiert Instanzdaten aus *trigger* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqString applicationId( ) const ;
Gibt eine Kopie der Anwendungs-ID (**application id**) zurück.

void setApplicationId(const char * *id*);
Legt die Anwendungs-ID (**application id**) fest.

MQLONG applicationType( ) const ;
Gibt den Anwendungstyp (**application type**) zurück.

void setApplicationType(const MQLONG *type*);
Legt den Anwendungstyp (**application type**) fest.

ImqBoolean copyOut(MQTMC2 * *ptmc2*);
Bindet die MQTM-Datenstruktur ein, die von den Initialisierungswarteschlangen empfangen wird. Gibt eine funktional entsprechende MQTMC2-Datenstruktur an, die von der aufrufenden Methode bereitgestellt wird, und legt für das gesamte Feld "QMgrName" (das in der MQTM-Datenstruktur nicht vorhanden ist) Leerzeichen fest. Die MQTMC2-Datenstruktur wird üblicherweise als Parameter für Anwendungen verwendet, die von einem Auslösemonitor gestartet werden. Bei erfolgreicher Ausführung dieser Methode wird TRUE zurückgegeben.

ImqString environmentData( ) const ;
Gibt eine Kopie der Umgebungsdaten (**environment data**) zurück.

void setEnvironmentData(const char * *data*);
Legt die Umgebungsdaten (**environment data**) fest.

ImqString processName( ) const ;
Gibt eine Kopie des Prozessnamens (**process name**) zurück.

void setProcessName(const char * *name*);
Legt den Prozessnamen (**process name**) fest, der mit Leerzeichen auf 48 Zeichen aufgefüllt wird.

ImqString queueName( ) const ;
Gibt eine Kopie des Warteschlangennamens (**queue name**) zurück.

void setQueueName(const char * *name*);
Legt den Warteschlangennamen (**queue name**) fest, der mit Leerzeichen auf 48 Zeichen aufgefüllt wird.

ImqString triggerData( ) const ;
Gibt eine Kopie der Auslöserdaten (**trigger data**) zurück.

void setTriggerData(const char * *data*);
Legt die Auslöserdaten (**trigger data**) fest.

ImqString userData( ) const ;
Gibt eine Kopie der Benutzerdaten (**user data**) zurück.

void setUserData(const char * *data*);
Legt die Benutzerdaten (**user data**) fest.

Objektdaten (geschützt)

MQTM *omqtm*
Die MQTM-Datenstruktur.

Ursachencodes

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

C++-Klasse ImqWorkHeader

Diese Klasse bindet bestimmte Komponenten der MQWIH-Datenstruktur ein.

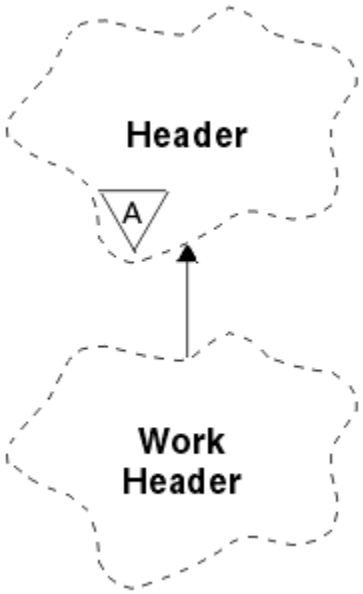


Abbildung 69. ImqWorkHeader-Klasse

Objekte dieser Klasse werden von Anwendungen verwendet, die Nachrichten in die vom z/OS Workload Manager verwaltete Warteschlange einreihen.

- „Objektattribute” auf Seite 1456
- „Konstruktoren” auf Seite 1457
- „Methoden für überlastete ImqItem-Klassen” auf Seite 1457
- „Objektmethoden (öffentlich)” auf Seite 1457
- „Objektdaten (geschützt)” auf Seite 1458
- „Ursachencodes” auf Seite 1458

Objektattribute

Nachrichten-Token
Nachrichtentoken für den z/OS Workload Manager mit der Länge MQ_MSG_TOKEN_LENGTH. Der Anfangswert lautet MQMTOK_NONE.

Servicename
Der aus 32 Zeichen bestehende Name eines Prozesses. Der Name besteht anfangs aus Leerzeichen.

service step
Der aus 8 Zeichen bestehende Name eines Schritts im Prozess. Der Name besteht anfangs aus Leerzeichen.

Konstruktoren

ImqWorkHeader( );
Der Standardkonstruktor.

ImqWorkHeader (const ImqWorkHeader & *Header*);
Der Kopierkonstruktor.

Methoden für überlastete ImqItem-Klassen

virtuelle ImqBoolean copyOut(ImqMessage & *msg*);
Fügt eine MQWIH-Datenstruktur am Anfang des Nachrichtenpuffers ein, verschiebt dabei die bereits vorhandenen Nachrichtendaten und setzt das *msg*-**Format** auf MQFMT_WORK_INFO_HEADER.

Weitere Details finden Sie in der Beschreibung der übergeordneten Klassenmethoden.

virtuelle ImqBoolean pasteIn(ImqMessage & *Nachricht*);
Liest eine MQWIH-Datenstruktur aus dem Nachrichtenpuffer.

Damit dieser Vorgang erfolgreich ist, muss als Codierung des *msg*-Objekts MQENC_NATIVE verwendet werden. Abrufen von Nachrichten mit MQGMO_CONVERT nach MQENC_NATIVE.

Das ImqMessage-Format muss MQFMT_WORK_INFO_HEADER lauten.

Weitere Details finden Sie in der Beschreibung der übergeordneten Klassenmethoden.

Objektmethoden (öffentlich)

void operator = (const ImqWorkHeader & *Header*);
Kopiert Instanzdaten aus *header* und ersetzt dabei die bereits vorhandenen Instanzdaten.

ImqBinary messageToken ( ) const;
Gibt das Nachrichtentoken (**message token**) zurück.

ImqBoolean setMessageToken (const ImqBinary & *Token*);
Legt das Nachrichtentoken (**message token**) fest. Die Datenlänge von *token* muss entweder null oder MQ_MSG_TOKEN_LENGTH betragen. Bei erfolgreicher Ausführung wird TRUE zurückgegeben.

void setMessageToken(const MQBYTE16 *token* = 0);
Legt das Nachrichtentoken (**message token**) fest. *token* darf null sein, was einer Festlegung von MQMTOK_NONE entspricht. Wenn *token* ungleich null ist, müssen Bytes mit einer Länge von MQ_MSG_TOKEN_LENGTH an binären Daten adressiert werden.

Wenn Sie vordefinierte Werte wie MQMTOK_NONE verwenden, müssen Sie möglicherweise eine Umsetzung vornehmen, um eine Signaturübereinstimmung sicherzustellen. Beispiel: (MQBYTE *)MQMTOK_NONE.

ImqString serviceName ( ) const;
Gibt den Servicenamen (**service name**) einschließlich abschließender Leerzeichen zurück.

void setServiceName(const char * *name*);
Legt den Servicenamen (**service name**) fest.

ImqString serviceStep ( ) const;
Gibt den Serviceschritt (**service step**) einschließlich abschließender Leerzeichen zurück.

void setServiceStep(const char * *step*);
Legt den Serviceschritt (**service step**) fest.

Objektdaten (geschützt)

MQWIH omqwih
Die MQWIH-Datenstruktur.

Ursachencodes

- MQRC_BINARY_DATA_LENGTH_ERROR

IBM WebSphere MQ -Klassen für Java-Bibliotheken

Die Position der IBM WebSphere MQ -Klassen für Java-Bibliotheken variiert je nach Plattform. Geben Sie diesen Standort an, wenn Sie eine Anwendung starten.

Zur Angabe der Position der JNI-Bibliotheken (JNI = Java Native Interface) starten Sie Ihre Anwendung mit einem **java** -Befehl im folgenden Format:

```
java -Djava.library.path=library_path application_name
```

Dabei ist *Bibliothekspfad* der Pfad zu den Bibliotheken von WebSphere MQ Classes for Java, die die JNI-Bibliotheken enthalten. Tabelle 615 auf Seite 1458 zeigt die Position der Bibliotheken von WebSphere MQ Classes for Java für jede Plattform.

Tabelle 615. Die Position der Bibliotheken von WebSphere MQ Classes for Java für jede Plattform

Plattform	**Verzeichnis mit den Bibliotheken der WebSphere MQ -Klassen für Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (32-Bit-Bibliotheken) *MQ_INSTALLATION_PATH*/java/lib64 (64-Bit-Bibliotheken)
HP-UX Linux (POWER, x86-64 und zSeries s390x -Plattformen) Solaris (x86-64- und SPARC-Plattformen)	*MQ_INSTALLATION_PATH*/java/lib (32-Bit-Bibliotheken) *MQ_INSTALLATION_PATH*/java/lib64 (64-Bit-Bibliotheken)
Linux (x86-Plattform)	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\Java\lib (32-Bit-Bibliotheken) *MQ_INSTALLATION_PATH*\Java\lib64 (64-Bit-Bibliotheken)
MQ_INSTALLATION_PATH steht für das übergeordnete Verzeichnis, in dem WebSphere MQ installiert ist.	

Anmerkung:

1. Verwenden Sie unter AIX, HP-UX, Linux (Power-Plattform) oder Solaris entweder die 32-Bit-Bibliotheken oder die 64-Bit-Bibliotheken. Verwenden Sie die 64-Bit-Bibliotheken nur, wenn Sie Ihre Anwendung in einer 64-Bit-Java Virtual Machine (JVM) auf einer 64-Bit-Plattform ausführen. Verwenden Sie ansonsten die 32-Bit-Bibliotheken.
2. Unter Windowskönnen Sie die Umgebungsvariable PATH verwenden, um die Position der Bibliotheken der WebSphere MQ -Klassen für Java anzugeben, anstatt sie im Befehl **java** anzugeben.

3. Wenn Sie WebSphere MQ Classes for Java im Bindungsmodus unter IBM iverwenden möchten, stellen Sie sicher, dass die Bibliothek QMQMJAVA in Ihrer Bibliotheksliste enthalten ist.

Zugehörige Tasks

WebSphere MQ Classes for Java verwenden

Eigenschaften der IBM WebSphere MQ classes for JMS-Objekte

Alle Objekte in IBM WebSphere MQ classes for JMS verfügen über Eigenschaften. Für die verschiedenen Objekttypen gelten jeweils unterschiedliche Eigenschaften. Verschiedene Eigenschaften haben verschiedene zulässige Werte, und die symbolischen Eigenschaftswerte unterschieden sich zwischen dem Verwaltungstool und dem Programmcode.

IBM WebSphere MQ classes for JMS stellt Funktionen für die Festlegung und Abfrage der Objekteigenschaften über das JMS-Verwaltungstool von WebSphere MQ, WebSphere MQ Explorer oder in einer Anwendung bereit. Viele der Eigenschaften sind nur für eine bestimmte Untergruppe der Objekttypen relevant.

Informationen zur Verwendung des JMS-Verwaltungstools für WebSphere MQ finden Sie unter Verwendung des JMS-Verwaltungstools für WebSphere MQ.

Tabelle 616 auf Seite 1459 enthält eine Kurzbeschreibung der einzelnen Eigenschaften und gibt für jede Eigenschaft an, für welche Objekttypen sie gilt. Die Objekttypen werden mithilfe von Schlüsselwörtern identifiziert; unter JMS-Objekttypen werden diese erläutert.

Die Nummern beziehen sich auf Hinweise am Ende der Tabelle. Weitere Informationen hierzu finden Sie im Abschnitt „Abhängigkeiten zwischen Eigenschaften von Objekten von WebSphere MQ Classes for JMS” auf Seite 1511.

Eine Eigenschaft besteht aus einem Name/Wert-Paar in folgendem Format:

```
PROPERTY_NAME(property_value)
```

Die Themen in diesem Abschnitt listen für jede Eigenschaft den Namen der Eigenschaft und eine Kurzbeschreibung auf. Außerdem werden die gültigen Eigenschaftswerte angegeben, die im Verwaltungstool verwendet werden. Darüber hinaus wird die Set-Methode angegeben, die zur Festlegung des Eigenschaftswerts in einer Anwendung genutzt wird. Neben diesen Informationen enthalten die Themen auch die gültigen Eigenschaftswerte für die einzelnen Eigenschaften und die Zuordnung zwischen den symbolischen Eigenschaftswerten, die im Tool und ihren programmierbaren Entsprechungen verwendet werden.

Die Groß- und Kleinschreibung spielt bei den Eigenschaftsnamen keine Rolle und die Namen sind auf die Gruppe erkannter Namen, die in diesen Themen genannt werden, beschränkt.

Tabelle 616. Eigenschaftsnamen und anwendbare Objekttypen

Eigenschaft	**Kurzform**	**Objekttyp**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
„APPLICATIONNAME” auf Seite 1463	APPNAME	Y	Y	Y			Y	Y	Y
„ASYNCEXCEPTION” auf Seite 1463	AEX	Y	Y	Y			Y	Y	Y
„BROKERCCDURSUBQ” auf Seite 1464[1]	CCDSUB					Y			
„BROKERCCSUBQ” auf Seite 1465[1]	CCSUB	Y		Y			Y		Y
„BROKERCONQ” auf Seite 1465[1]	BCON	Y		Y			Y		Y
„BROKERDURSUBQ” auf Seite 1466[1]	BDSUB					Y			
„BROKERPUBQ” auf Seite 1466[1]	BPUB	Y		Y		Y	Y		Y
„BROKERPUBQMGR” auf Seite 1466[1]	BPQM					Y			

Tabelle 616. Eigenschaftsnamen und anwendbare Objekttypen (Forts.)

Eigenschaft	Kurzform	Objekttyp							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
„BROKERQMGR” auf Seite 1467[1]	BQM	Y		Y			Y		Y
„BROKERSUBQ” auf Seite 1467[1]	BSUB	Y		Y			Y		Y
„BROKERVER” auf Seite 1468[1]	BVER	Y[2]		Y[2]		Y	Y		Y
„CCDTURL” auf Seite 1468[3]	CCDT (Client Channel Definition Table)	Y	Y	Y			Y	Y	Y
„CCSID” auf Seite 1469	CCS	Y	Y	Y	Y	Y	Y	Y	Y
„CHANNEL” auf Seite 1469[3]	CHAN	Y	Y	Y			Y	Y	Y
„CLEANUP” auf Seite 1470[1]	CL	Y		Y			Y		Y
„CLEANUPINT” auf Seite 1470[1]	CLINT	Y		Y			Y		Y
„CONNECTIONNAMELIST” auf Seite 1471	CNLIST	Y	Y	Y					
„CLIENTRECONNECTOPTIONS” auf Seite 1471	CROPT	Y	Y	Y					
„CLIENTRECONNECTTIMEOUT” auf Seite 1472	CRT	Y	Y	Y					
„CLIENTID” auf Seite 1473	CID	Y[2]	Y	Y[2]			Y	Y	Y
„CLONESUPP” auf Seite 1473	CLS	Y		Y			Y		Y
„COMPHDR” auf Seite 1474	HC	Y		Y			Y		Y
„COMPMSG” auf Seite 1474	MC	Y	Y	Y			Y	Y	Y
„CONNOPT” auf Seite 1474	CNOPT	Y	Y	Y			Y	Y	Y
„CONNTAG” auf Seite 1476	CNTAG	Y	Y	Y			Y	Y	Y
„DESCRIPTION” auf Seite 1476	DESC	Y[2]	Y	Y[2]	Y	Y	Y	Y	Y
„DIRECTAUTH” auf Seite 1476	DAUTH	Y[2]		Y[2]					
„ENCODING” auf Seite 1477	ENC				Y	Y			
„EXPIRY” auf Seite 1478	EXP				Y	Y			
„FAILIFQUIESCE” auf Seite 1478	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
„HOSTNAME” auf Seite 1479	HOST	Y[2]	Y	Y[2]			Y	Y	Y
„LOCALADDRESS” auf Seite 1480	LA	Y[2]	Y	Y[2]			Y	Y	Y
„MAPNAMESTYLE” auf Seite 1481	MNST	Y	Y	Y			Y	Y	Y
„MAXBUFFSIZE” auf Seite 1481	MBSZ	Y[2]		Y[2]					
„MDREAD” auf Seite 1482	MDR				Y	Y			
„MDWRITE” auf Seite 1482	MDW				Y	Y			
„MDMSGCTX” auf Seite 1483	MDCTX				Y	Y			

Tabelle 616. Eigenschaftsnamen und anwendbare Objekttypen (Forts.)

Eigenschaft	Kurzform	Objekttyp							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
„MSGBATCHSZ” auf Seite 1483[1]	MBS	Y	Y	Y			Y	Y	Y
„MSGBODY” auf Seite 1484	MBODY				Y	Y			
„MSGRETENTION” auf Seite 1484	MRET	Y	Y				Y	Y	
„MSGSELECTION” auf Seite 1485[1]	MSEL	Y		Y			Y		Y
„MULTICAST” auf Seite 1485	MCAST	Y[2]		Y[2]		Y			
„OPTIMISTICPUBLICATION” auf Seite 1486[1]	OPTPUB	Y		Y					
„OUTCOMENOTIFICATION” auf Seite 1487[1]	NOTIFY	Y		Y					
„PERSISTENCE” auf Seite 1487	PER				Y	Y			
„POLLINGINT” auf Seite 1488[1]	PINT	Y	Y	Y			Y	Y	Y
„PORT” auf Seite 1488	PORT	Y[2]	Y	Y[2]			Y	Y	Y
„PRIORITY” auf Seite 1489	PRI				Y	Y			
„PROCESSDURATION” auf Seite 1489[1]	PROCDUR	Y		Y					
„PROVIDERVERSION” auf Seite 1490	PVER	Y	Y	Y			Y	Y	Y
„PROXYHOSTNAME” auf Seite 1491	PHOST	Y[2]		Y[2]					
„PROXYPORT” auf Seite 1492	PPORT	Y[2]		Y[2]					
„PUBACKINT” auf Seite 1492[1]	PAI	Y		Y			Y		Y
„PUTASYNCALLOWED” auf Seite 1493	PAALD				Y	Y			
„QMANAGER” auf Seite 1493	QMGR	Y	Y	Y	Y		Y	Y	Y
„WARTESCHLANGE” auf Seite 1494	QU				Y				
„READAHEADALLOWED” auf Seite 1494	RAALD				Y	Y			
„READAHEADCLOSEPOLICY” auf Seite 1495	RACP				Y	Y			
„RECEIVECCSID” auf Seite 1496	RCCS				Y	Y			
„RECEIVECONVERSION” auf Seite 1496	RCNV				Y	Y			
„RECEIVEISOLATION” auf Seite 1497[1]	RCVISOL	Y		Y					
„RECEXIT” auf Seite 1497	RCX	Y	Y	Y			Y	Y	Y
„RECEXITINIT” auf Seite 1498	RCXI	Y	Y	Y			Y	Y	Y
„REPLYTOSTYLE” auf Seite 1498	RTOST				Y	Y			
„RESCANINT” auf Seite 1499[1]	RINT	Y	Y				Y	Y	
„SECEXIT” auf Seite 1499	SCX	Y	Y	Y			Y	Y	Y

Tabelle 616. Eigenschaftsnamen und anwendbare Objekttypen (Forts.)

Eigenschaft	Kurzform	Objekttyp							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
„SECEXITINIT” auf Seite 1500	SCXI	Y	Y	Y			Y	Y	Y
„SENDCHECKCOUNT” auf Seite 1500	SCC	Y	Y	Y			Y	Y	Y
„SENDEXIT” auf Seite 1501	SDX	Y	Y	Y			Y	Y	Y
„SENDEXITINIT” auf Seite 1501	SDXI	Y	Y	Y			Y	Y	Y
„SHARECONVALLOWED” auf Seite 1502	SCALD	Y	Y	Y			Y	Y	Y
„SPARSESUBS” auf Seite 1502[1]	SSUBS	Y		Y					
„SSLCIPHERSUITE” auf Seite 1503	SCPHS	Y	Y	Y			Y	Y	Y
„SSLCRL” auf Seite 1503	SCRL	Y	Y	Y			Y	Y	Y
„SSLFIPSREQUIRED” auf Seite 1504	SFIPS	Y	Y	Y			Y	Y	Y
„SSLPEERNAME” auf Seite 1504	SPEER	Y	Y	Y			Y	Y	Y
„SSLRESETCOUNT” auf Seite 1505	SRC	Y	Y	Y			Y	Y	Y
„STATREFRESHINT” auf Seite 1505[1]	SRI	Y		Y			Y		Y
„SUBSTORE” auf Seite 1506[1]	SS	Y		Y			Y		Y
„SYNCPOINTALLGETS” auf Seite 1506	SPAG	Y	Y	Y			Y	Y	Y
„TARGCLIENT” auf Seite 1507	TC				Y	Y			
„TARGCLIENTMATCHING” auf Seite 1507	TCM	Y	Y				Y	Y	
„TEMPMODEL” auf Seite 1508	TM	Y	Y				Y	Y	
„TEMPQPREFIX” auf Seite 1508	TQP	Y	Y				Y	Y	
„TEMPTOPICPREFIX” auf Seite 1509	TTP	Y		Y			Y		Y
„TOPIC” auf Seite 1509	TOP					Y			
„TRANSPORT” auf Seite 1509	TRAN	Y[2]	Y	Y[2]			Y	Y	Y
„WILDCARDFORMAT” auf Seite 1510	WCFMT	Y		Y			Y		Y

Anmerkung:

1. Diese Eigenschaft kann in Verbindung mit Version 7.0 von WebSphere MQ-Klassen für Java Message Service verwendet werden, hat jedoch bei einer Anwendung, die mit einem Warteschlangenmanager der Version 7.0 verbunden ist, keine Auswirkung, es sei denn, die Eigenschaft PROVIDERVERSION der Verbindungsfactory wird auf eine niedrigere Versionsnummer als 7 gesetzt.
2. Wenn eine Echtzeitverbindung mit einem Broker genutzt wird, werden für ein ConnectionFactory- oder TopicConnectionFactory-Objekt nur die Eigenschaften BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT und TRANSPORT unterstützt.
3. Die Eigenschaften CCDTURL und CHANNEL eines Objekts dürfen nicht gleichzeitig festgelegt werden.

APPLICATIONNAME

Eine Anwendung kann einen Namen festlegen, der die Verbindung zum Warteschlangenmanager angibt. Dieser Anwendungsname wird durch den Befehl **DISPLAY CONN MQSC/PCF** (mit dem Feld **APPLTAG**) oder in der Anzeige **Anwendungsverbindungen** von IBM WebSphere MQ Explorer (mit dem Feld **App name**) angezeigt.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: APPLICATIONNAME

Kurzname des JMS-Verwaltungstools: APPNAME

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setAppName()
- MQConnectionFactory.getAppName()

Werte

Jede gültige Zeichenfolge mit nicht mehr als 28 Zeichen. Längere Namen werden bei Bedarf angepasst, indem am Anfang beginnend die Paketnamen entfernt werden. Wenn zum Beispiel `com.example.MainApp` die aufrufende Klasse ist, wird der vollständige Name verwendet, wenn aber `com.example.dictionaryAndThesaurus.multilingual.mainApp` die aufrufende Klasse ist, wird der Name `multilingual.mainApp` verwendet, da dies die längste Kombination aus Klassenname und, von rechts gesehen, dem Paketnamen ist, mit dem die Längenvorgabe erfüllt ist.

Wenn der Klassenname selbst mehr als 28 Zeichen lang ist, wird er entsprechend abgeschnitten. Zum Beispiel wird statt `com.example.mainApplicationForSecondTestCase` die Zeichenfolge `mainApplicationForSecondTest` verwendet.

ASYNCEXCEPTION

Diese Eigenschaft bestimmt, ob WebSphere MQ Classes for JMS nur dann einen ExceptionListener informiert, wenn eine Verbindung unterbrochen wird oder auch dann, wenn eine beliebige Ausnahmebedingung asynchron zu einem JMS-API-Aufruf auftritt. Dies gilt für alle Connections (Verbindungen), die aus dieser ConnectionFactory erstellt wurden, für die ein ExceptionListener registriert ist.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: ASYNCEXCEPTION

Kurzname des JMS-Verwaltungstools: AEX

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setAsyncExceptions()
- MQConnectionFactory.getAsyncExceptions()

Werte

ASYNC_EXCEPTIONS_ALL

Alle Ausnahmebedingungen, die asynchron außerhalb des Bereichs eines synchronen API-Aufrufs erkannt wurden, und alle Ausnahmebedingungen durch unterbrochene Verbindungen werden an den ExceptionListener gesendet.

Umgebung	Wert
JMS-Verwaltungstool	ALLE
Programmgesteuert	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
WebSphere MQ Explorer	Alle

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Nur Ausnahmebedingungen, die eine unterbrochene Verbindung angeben, werden an den ExceptionListener gesendet. Alle anderen Ausnahmebedingungen, die während der asynchronen Verarbeitung auftreten, werden nicht an den ExceptionListener gemeldet, d. h. die Anwendung wird über diese Ausnahmebedingungen nicht informiert. V 7.5.0.8 Dies ist ab IBM WebSphere MQ Version 7.5.0Fixpack 8 der Standardwert (siehe JMS: Änderungen am Listener für Ausnahmebedingungen in Version 7.5).

Umgebung	Wert
JMS-Verwaltungstool	CONNECTIONBROKEN
Programmgesteuert	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
WebSphere MQ Explorer	Verbindung unterbrochen

Die folgende zusätzliche Konstante wird definiert: V 7.5.0.8

- Ab Version 7.5.0, Fix Pack 8: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- Vor Version 7.5.0, Fix Pack 8: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Zugehörige Konzepte

Ausnahmebedingungen in WebSphere MQ-Klassen für JMS

BROKERCCDURSUBQ

Der Name der Warteschlange, aus der permanente Subskriptionsnachrichten für einen ConnectionConsumer (Verbindungskonsumenten) abgerufen werden.

Gültige Objekte

Thema

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERCCDURSUBQ

Kurzname des JMS-Verwaltungstools: CCDSUB

Programmgesteuerter Zugriff

Setter/Getter

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Werte

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERCCSUBQ

Der Name der Warteschlange, aus der nicht permanente Subskriptionsnachrichten für einen Connection-Consumer (Verbindungskonsumenten) abgerufen werden.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERCCSUBQ

Kurzname des JMS-Verwaltungstools: CCSUB

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Werte

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERCONQ

Der Steuerwarteschlangenname des Brokers.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERCONQ

Kurzname des JMS-Verwaltungstools: BCON

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Werte

SYSTEM.BROKER.CONTROL.QUEUE
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERDURSUBQ

Bei Verwendung der WebSphere MQ-Klassen für JMS im WebSphere MQ-Migrationsmodus für Messaging-Provider gibt diese Eigenschaft den Namen der Warteschlange an, aus der permanente Subskriptionsnachrichten abgerufen werden.

Gültige Objekte

Thema

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERDURSUBQ

Kurzname des JMS-Verwaltungstools: BDSUB

Programmgesteuerter Zugriff

Setter/Getter

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Werte

SYSTEM.JMS.D.SUBSCRIBER.QUEUE
Dies ist der Standardwert.

Jede gültige Zeichenfolge.
Mit SYSTEM.JMS.D beginnend.

Zugehörige Konzepte
Regeln zur Auswahl des WebSphere MQ-Modus für Messaging-Provider

BROKERPUBQ

Der Name der Warteschlange, an die veröffentlichte Nachrichten gesendet werden (die Datenstromwarteschlange).

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERPUBQ

Kurzname des JMS-Verwaltungstools: BPUB

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Werte

SYSTEM.BROKER.DEFAULT.STREAM
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERPUBQMGR

Der Name des Warteschlangenmanagers, in dem die Warteschlange definiert ist, an die die zu diesem Thema veröffentlichten Nachrichten gesendet werden.

Gültige Objekte

Thema

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERPUBQMGR

Kurzname des JMS-Verwaltungstools: BPQM

Programmgesteuerter Zugriff

Setter/Getter

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Werte

null
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERQMGR

Der Namen des Warteschlangenmanagers, für den der Broker aktiv ist.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERQMGR

Kurzname des JMS-Verwaltungstools: BQM

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Werte

null
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

BROKERSUBQ

Bei Verwendung der WebSphere MQ-Klassen für JMS im WebSphere MQ-Migrationsmodus für Messaging-Provider gibt diese Eigenschaft den Namen der Warteschlange an, aus der nicht permanente Subskriptionsnachrichten abgerufen werden.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERSUBQ

Kurzname des JMS-Verwaltungstools: BSUB

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Werte

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
Dies ist der Standardwert.

Jede gültige Zeichenfolge.
Mit SYSTEM.JMS.ND beginnend.

Zugehörige Konzepte
Regeln zur Auswahl des WebSphere MQ-Modus für Messaging-Provider

BROKERVER

Die Version des verwendeten Brokers.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: BROKERVER

Kurzname des JMS-Verwaltungstools: BVER

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setBrokerVersion()
- MQConnectionFactory.getBrokerVersion()

Werte

V1
Um einen WebSphere MQ-Publish/Subscribe-Broker oder um einen Broker von WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker oder einen WebSphere Business Integration Message Broker im Kompatibilitätsmodus zu verwenden. Dies ist der Standardwert, wenn für TRANSPORT der Wert BIND oder CLIENT festgelegt ist.

V2
Um einen Broker von WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker oder einen WebSphere Business Integration Message Broker im nativen Modus zu verwenden. Dies ist der Standardwert, wenn für TRANSPORT der Wert DIRECT oder DIRECTHTTP festgelegt ist.

nicht angegeben
Legen Sie diese Eigenschaft so fest, dass RFH2-Header nicht mehr verwendet werden, nachdem der Broker von V6 zu V7 migriert wurde. Nach der Migration ist diese Eigenschaft nicht mehr relevant.

CCDTURL

Eine URL, die den Namen und die Position der Datei bestimmt, die die Kanaldefinitionstabelle des Clients enthält und festlegt, wie auf die Datei zugegriffen werden kann.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CCDTURL

Kurzname des JMS-Verwaltungstools: CCDT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Werte

null
Dies ist der Standardwert.

Eine URL.

CCSID

Die ID des codierten Zeichensatzes, die für eine Verbindung oder ein Ziel verwendet werden soll.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CCSID

Kurzname des JMS-Verwaltungstools: CCS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Werte

819
Dies ist der Standardwert für eine Verbindungsfactory.

1208
Dies ist der Standardwert für ein Ziel.

Jede beliebige positive Ganzzahl.

CHANNEL

Der Name des verwendeten Clientverbindungskanals.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CHANNEL

Kurzname des JMS-Verwaltungstools: CHAN

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Werte

SYSTEM.DEF.SVRCONN
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

CLEANUP

Bereinigungsstufe für BROKER- oder MIGRATE-Subskriptionsspeicher.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLEANUP

Kurzname des JMS-Verwaltungstools: CL

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setCleanupLevel()
- MQConnectionFactory.getCleanupLevel()

Werte

SAFE
Sichere Bereinigung verwenden. Dies ist der Standardwert.

ASPROP
Verwenden Sie sichere, starke oder keine Bereinigung gemäß einer Eigenschaft, die in der Java-Befehlszeile festgelegt wurde.

KEINE
Keine Bereinigung verwenden.

STRONG
Starke Bereinigung verwenden.

CLEANUPINT

Das Intervall in Millisekunden zwischen den Ausführungen des Publish/Subscribe-Bereinigungsdienstprogramms im Hintergrund.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLEANUPINT

Kurzname des JMS-Verwaltungstools: CLINT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setCleanupInterval()
- MQConnectionFactory.getCleanupInterval()

Werte

3600000
Dies ist der Standardwert.

Jede beliebige positive Ganzzahl.

CONNECTIONNAMELIST

Liste der TCP/IP-Verbindungsnamen. Die Verbindungsadressen in der Liste werden der Reihe nach ausprobiert und zwar einmal für jeden Verbindungswiederholungsversuch.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CONNECTIONNAMELIST

Kurzname des JMS-Verwaltungstools: CNLIST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setconnectionNameList()
- MQConnectionFactory.getconnectionNameList()

Werte

Durch Kommas getrennte Liste von HOSTNAME(PORT). HOSTNAME kann entweder ein DNS-Name oder eine IP-Adresse sein.

PORT hat den Standardwert "1414".

CLIENTRECONNECTOPTIONS

Die Optionen, die die Verbindungswiederholungen regeln.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLIENTRECONNECTOPTIONS

Kurzname des JMS-Verwaltungstools: CROPT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Werte

QMGR

Die Anwendung kann eine Verbindung wiederherstellen, jedoch nur zu dem Warteschlangenmanager, zu dem sie ursprünglich eine Verbindung hergestellt hat.

Verwenden Sie diesen Wert, wenn die Verbindung zu einer Anwendung wiederhergestellt werden kann, jedoch eine Affinität zwischen der WebSphere MQ Classes for JMS-Anwendung und dem Warteschlangenmanager besteht, zu dem zuerst eine Verbindung hergestellt wurde.

Geben Sie diesen Wert an, wenn eine Anwendung automatisch wieder mit der Standby-Instanz eines hochverfügbaren Warteschlangenmanagers verbunden werden soll.

Wenn Sie diesen Wert programmgesteuert nutzen möchten, verwenden Sie die Konstante WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

ANY

Die Anwendung kann die Verbindung zu einem beliebigen Warteschlangenmanager wiederherstellen.

Verwenden Sie die Option zur Wiederherstellung einer Verbindung nur, wenn zwischen der WebSphere MQ Classes for JMS-Anwendung und dem Warteschlangenmanager, mit dem ursprünglich eine Verbindung hergestellt wurde, keine Affinität besteht.

Wenn Sie diesen Wert aus einem Programm nutzen möchten, verwenden Sie die Konstante WMQConstants.WMQ_CLIENT_RECONNECT.

INAKTIVIERT

Die Anwendung wird nicht wieder verbunden.

Wenn Sie diesen Wert programmgesteuert nutzen möchten, verwenden Sie die Konstante WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

Es hängt vom Wert des WebSphere MQ-Kanalattributs 'DefReconnect' ab, ob die Verbindung zu der Anwendung automatisch wiederhergestellt wird.

Wenn Sie diesen Wert aus einem Programm nutzen möchten, verwenden Sie die Konstante WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Die Zeit, bevor die Verbindungswiederholungsversuche eingestellt werden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLIENTRECONNECTTIMEOUT

Kurzname des JMS-Verwaltungstools: CRT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Werte

Intervall in Sekunden. Der Standardwert lautet "1800" (30 Minuten).

CLIENTID

Die Client-ID wird verwendet, um die Anwendungsverbindung für permanente Subskriptionen eindeutig zu identifizieren.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-ConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLIENTID

Kurzname des JMS-Verwaltungstools: CID

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setClientId()
- MQConnectionFactory.getClientId()

Werte

null
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

CLONESUPP

Geben Sie an, ob zwei oder mehr Instanzen desselben permanenten Topic-Subskribenten gleichzeitig ausgeführt werden können.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CLONESUPP

Kurzname des JMS-Verwaltungstools: CLS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setCloneSupport()
- MQConnectionFactory.getCloneSupport()

Werte

INAKTIVIERT
Es kann jeweils nur eine Instanz eines Subskribenten für permanente Themen ausgeführt werden. Dies ist der Standardwert.

ENABLED
Zwei oder mehr Instanzen desselben permanenten Topic-Subskribenten können gleichzeitig ausgeführt werden, aber jede Instanz muss in einer separaten Java Virtual Machine (JVM) ausgeführt werden.

COMPHDR

Eine Liste der Verfahren, die zum Komprimieren von Headerdaten in einer Verbindung verwendet werden können.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: COMPHDR

Kurzname des JMS-Verwaltungstools: HC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Werte

KEINE
Dies ist der Standardwert.

SYSTEM
Es wird eine Komprimierung des RLE-Nachrichtenheaders durchgeführt.

COMPMSG

Eine Liste der Verfahren, die zum Komprimieren von Nachrichtendaten in einer Verbindung verwendet werden können.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-ConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: COMPMSG

Kurzname des JMS-Verwaltungstools: MC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Werte

KEINE
Dies ist der Standardwert.

Eine Liste mit einem oder mehreren der folgenden Werte, die durch Leerzeichen voneinander getrennt sind:
RLE ZLIBFAST ZLIBHIGH

CONNOPT

Steuert, wie WebSphere MQ-Klassen für JMS-Anwendungen, die den Bindungstransport verwenden, eine Verbindung mit dem Warteschlangenmanager herstellen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-ConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CONNOPT

Kurzname des JMS-Verwaltungstools: CNOPT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMQConnectionOptions()
- MQConnectionFactory.getMQConnectionOptions()

Werte

STANDARD
Die Art der Bindung zwischen der Anwendung und dem Warteschlangenmanager ist vom Wert des Attributs *DefaultBindType* des Warteschlangenmanagers abhängig. Der Wert STANDARD entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_STANDARD_BINDING

SHARED
Die Anwendung und der lokale Warteschlangenmanageragent werden in unterschiedlichen Ausführungseinheiten ausgeführt, nutzen jedoch einige Ressourcen gemeinsam. Dieser Wert entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_SHARED_BINDING.

ISOLATED
Die Anwendung und der lokale Warteschlangenmanageragent werden in unterschiedlichen Ausführungseinheiten ausgeführt und nutzen keine Ressourcen gemeinsam. Der Wert ISOLATED entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_ISOLATED_BINDING.

FASTPATH
Die Anwendung und der lokale Warteschlangenmanageragent werden in derselben Ausführungseinheit ausgeführt. Dieser Wert entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_FASTPATH_BINDING.

SERIALQM
Die Anwendung fordert eine exklusive Nutzung des Verbindungstags innerhalb des Bereichs des Warteschlangenmanagers an. Dieser Wert entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG
Die Anwendung fordert eine exklusive Nutzung des Verbindungstags innerhalb des Bereichs der Gruppe für gemeinsame Warteschlangennutzung, zu der der Warteschlangenmanager gehört. Der Wert SERIALQSG entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_SERIALIZE_CONN_TAG_QSG.

RESTRICTQM
Die Anwendung fordert eine gemeinsame Nutzung des Verbindungstags an, aber es gelten Einschränkungen für die gemeinsame Nutzung des Verbindungstags innerhalb des Bereichs des Warteschlangenmanagers. Dieser Wert entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

RESTRICTQSG
Die Anwendung fordert eine gemeinsame Nutzung des Verbindungstags an, aber es gelten Einschränkungen für die gemeinsame Nutzung des Verbindungstags innerhalb des Bereichs der Gruppe für gemeinsame Warteschlangennutzung, zu der der Warteschlangenmanager gehört. Dieser Wert entspricht der Zuordnung zur WebSphere MQ-*Verbindungsoption* MQCNO_RESTRICT_CONN_TAG_QSG.

Weitere Informationen zu WebSphere MQ-Verbindungsoptionen finden Sie im Abschnitt Verbindung zu einem Warteschlangenmanager über MQCONNX-Aufrufe herstellen

CONNTAG

Ein Tag, den der Warteschlangenmanager den Ressourcen zuordnet, die durch die Anwendung innerhalb einer Arbeitseinheit aktualisiert werden, während die Anwendung mit dem Warteschlangenmanager verbunden ist.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: CONNTAG

Kurzname des JMS-Verwaltungstools: CNTAG

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setConnTag()
- MQConnectionFactory.getConnTag()

Werte

Ein Byte-Array von 128 Elementen, wobei jedes Element "0" ist.
Dies ist der Standardwert.

Jede beliebige Zeichenfolge.
Der Wert wird abgeschnitten, wenn er länger als 128 Bytes ist.

DESCRIPTION

Eine Beschreibung des gespeicherten Objekts.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: DESCRIPTION

Kurzname des JMS-Verwaltungstools: DESC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Werte

null
Dies ist der Standardwert.

Jede gültige Zeichenfolge.
`

DIRECTAUTH

Ob eine SSL-Authentifizierung für eine Echtzeitverbindung zu einem Broker verwendet wird.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: DIRECTAUTH

Kurzname des JMS-Verwaltungstools: DAUTH

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setDirectAuth()
- MQConnectionFactory.getDirectAuth()

Werte

BASIC
Keine Authentifizierung, Benutzernamensauthentifizierung oder Kennwortauthentifizierung. Dies ist der Standardwert.

CERTIFICATE
Authentifizierung über ein Zertifikat für einen öffentlichen Schlüssel.

ENCODING

Wie numerische Daten im Hauptteil einer Nachricht dargestellt werden, wenn die Nachricht an das Ziel gesendet wird. Die Eigenschaft gibt die Darstellung von binären Ganzzahlen, gepackten Dezimalganzzahlen und Gleitkommazahlen an.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: ENCODING

Kurzname des JMS-Verwaltungstools: ENC

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Werte

Eigenschaft ENCODING

Die gültigen Werte, die die Eigenschaft ENCODING annehmen kann, werden aus den drei Untereigenschaften konstruiert:

Ganzzahlverschlüsselung
Normal oder umgekehrt

Dezimalverschlüsselung
Normal oder umgekehrt

Gleitkommaverschlüsselung
IEEE (normal), IEEE (umgekehrt) oder z/OS

Die Eigenschaft ENCODING wird als Zeichenfolge aus drei Zeichen mit der folgenden Syntax dargestellt:

```
{N|R}{N|R}{N|R|3}
```

Die Zeichen in dieser Zeichenfolge stehen dabei für Folgendes:

- N bedeutet "normal"
- R bedeutet "umgekehrt"
- 3 bedeutet z/OS
- Das erste Zeichen stellt eine *Ganzzahlverschlüsselung* dar
- Das zweite Zeichen stellt eine *Dezimalverschlüsselung* dar
- Das dritte Zeichen stellt eine *Gleitkommaverschlüsselung* dar

Diese Kombinationen stellen zwölf mögliche Werte für die Eigenschaft ENCODING bereit.

Es gibt einen zusätzlichen Wert, die Zeichenfolge NATIVE, die die entsprechenden Codierungswerte für die Java-Plattform festlegt.

In den folgenden Beispielen sind gültige Kombinationen für ENCODING dargestellt:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

EXPIRY

Die Zeit, nach der Nachrichten an ein Ziel ablaufen.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: EXPIRY

Kurzname des JMS-Verwaltungstools: EXP

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setExpiry()
- MQDestination.getExpiry()

Werte

APP
Die Ablaufzeit kann durch die JMS-Anwendung definiert werden. Dies ist der Standardwert.

UNLIM
Es gibt keine Ablaufzeit.

0
Es gibt keine Ablaufzeit.

Jede positive Ganzzahl, die die Ablaufzeit in Millisekunden darstellt.

FAILIFQUIESCE

Diese Eigenschaft bestimmt, ob Aufrufe bestimmter Methoden fehlschlagen, wenn sich der Warteschlangenmanager im Stilllegungsstatus befindet oder wenn eine Anwendung über den CLIENT-Transport eine Verbindung zu einem Warteschlangenmanager herstellt und der von der Anwendung verwendete Kanal

in den Stilllegungsstatus versetzt wurde, z. B. mit dem Befehl **STOP CHANNEL** oder dem MQSC **STOP CHANNEL MODE(QUIESCE)** .

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: FAILIFQUIESCE

Kurzname des JMS-Verwaltungstools: FIQ

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Werte

YES
Aufrufe bestimmter Methoden schlagen fehl, wenn sich der Warteschlangenmanager im Stilllegungsmodus befindet oder der Kanal, mit dem eine Verbindung zu einem Warteschlangenmanager hergestellt wird, stillgelegt wird. Wenn eine Anwendung eine dieser beiden Bedingungen erkennt, kann sie ihre aktuelle Task beenden und die Verbindung trennen, sodass der Warteschlangenmanager oder die Kanalinstanz gestoppt werden kann. Dies ist der Standardwert.

Nein
Keine der beiden Methoden schlägt fehl, da sich der Warteschlangenmanager oder der Kanal, mit dem eine Verbindung zu einem Warteschlangenmanager hergestellt wird, im Stilllegungsmodus befindet. Wenn Sie diesen Wert angeben, kann eine Anwendung nicht feststellen, ob sich der Warteschlangenmanager oder der Kanal im Stilllegungsmodus befindet. Die Anwendung fährt möglicherweise fort, Operationen für den Warteschlangenmanager auszuführen und verhindert damit, dass der Warteschlangenmanager anhält.

HOSTNAME

Für eine Verbindung zu einem Warteschlangenmanager der Hostname oder die IP-Adresse des Systems, auf dem der Warteschlangenmanager ausgeführt wird, oder für eine Echtzeitverbindung zu einem Broker der Hostname oder die IP-Adresse des Systems, auf dem der Broker ausgeführt wird.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: HOSTNAME

Kurzname des JMS-Verwaltungstools: HOST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setHostName()
- MQConnectionFactory.getHostName()

Werte

localhost
Dies ist der Standardwert.

Jede gültige Zeichenfolge.

LOCALADDRESS

Für eine Verbindung zu einem Warteschlangenmanager gibt diese Eigenschaft entweder die lokale Netzschnittstelle, die verwendet werden soll, den zu verwendenden lokalen Port oder den Bereich lokaler Ports an. Bei einer Echtzeitverbindung zu einem Broker ist diese Eigenschaft nur bei Verwendung von Multicasting relevant. Sie gibt die lokale Netzschnittstelle an, die verwendet werden soll.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: LOCALADDRESS

Kurzname des JMS-Verwaltungstools: LA

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setLocalAddress()
- MQConnectionFactory.getLocalAddress()

Werte

"" (leere Zeichenfolge)
Dies ist der Standardwert.

Eine Zeichenfolge im Format [ip-addr][(low-port[,high-port])]
Einige Beispiele:

192.0.2.0

Der Kanal wird lokal an die Adresse 192.0.2.0 gebunden.

192.0.2.0(1000)

Der Kanal wird lokal an die Adresse 192.0.2.0 gebunden und verwendet Port 1000.

192.0.2.0(1000,2000)

Der Kanal wird lokal an die Adresse 192.0.2.0 gebunden und verwendet einen Port im Bereich von 1000 bis 2000.

(1000)

Der Kanal wird lokal an Port 1000 gebunden.

(1000,2000)

Der Kanal wird lokal an einen Port im Bereich von 1000 bis 2000 gebunden.

Sie können einen Hostnamen anstelle einer IP-Adresse angeben. Bei einer Echtzeitverbindung zu einem Broker ist diese Eigenschaft nur bei Verwendung von Multicasting relevant. Der Wert der Eigenschaft darf keine Portnummer oder einen Bereich von Portnummern enthalten. Die einzigen gültigen Werte für die Eigenschaft sind in diesem Fall null, eine IP-Adresse oder ein Hostname.

MAPNAMESTYLE

Ermöglicht das Verwenden des Kompatibilitätsstils für MapMessage-Elementnamen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-ConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: MAPNAMESTYLE

Kurzname des JMS-Verwaltungstools: MNST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Werte

STANDARD
Das Standardformat für Elementnamen ("com.ibm.jms.JMSMapMessage") wird verwendet. Dies ist der Standardwert, der die Verwendung ungültiger Java-Bezeichner als Elementnamen ermöglicht.

COMPATIBLE
Das ältere Format für Elementnamen ("com.ibm.jms.JMSMapMessage") wird verwendet. Nur gültige Java-Bezeichner können als Elementnamen verwendet werden. Dies ist nur erforderlich, wenn Mapnachrichten an eine Anwendung gesendet werden, die eine ältere Version von IBM WebSphere MQ classes for JMS als Version 5.3 verwendet.

MAXBUFFSIZE

Die maximale Anzahl erhaltener Nachrichten, die in einem internen Nachrichtenpuffer gespeichert werden können, während auf die Verarbeitung der Nachrichten durch die Anwendung gewartet wird. Diese Eigenschaft wird nur verwendet, wenn für TRANSPORT der Wert DIRECT oder der Wert DIRECTHTTP festgelegt wurde.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: MAXBUFFSIZE

Kurzname des JMS-Verwaltungstools: MBSZ

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Werte

1.000
Dies ist der Standardwert.

Jede beliebige positive Ganzzahl.

MDREAD

Diese Eigenschaft bestimmt, ob eine JMS-Anwendung die Werte von MQMD-Feldern extrahieren kann.

Gültige Objekte

Ausgeschriebener Name des JMS-Verwaltungstools: MDREAD

Kurzname des JMS-Verwaltungstools: MDR

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Werte

Nein
Beim Senden von Nachrichten werden die JMS_IBM_MQMD*-Eigenschaften einer gesendeten Nachricht nicht mit den aktualisierten Feldwerten in der MQMD-Struktur aktualisiert. Beim Empfangen von Nachrichten ist keine der JMS_IBM_MQMD*-Eigenschaften in einer empfangenen Nachricht verfügbar, auch wenn der Absender einige oder alle dieser Eigenschaften festgelegt hatte. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie "False" für Programme.

Ja
Beim Senden von Nachrichten werden alle JMS_IBM_MQMD*-Eigenschaften einer gesendeten Nachricht mit den aktualisierten Feldwerten in der MQMD-Struktur aktualisiert, einschließlich der Eigenschaften, die der Absender nicht explizit festgelegt hat. Beim Empfangen von Nachrichten sind alle JMS_IBM_MQMD*-Eigenschaften in einer empfangenen Nachricht verfügbar, einschließlich der Eigenschaften, die der Absender nicht explizit festgelegt hat.

Verwenden Sie "True" für Programme.

MDWRITE

Diese Eigenschaft bestimmt, ob eine JMS-Anwendung die Werte von MQMD-Feldern festlegen kann.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: MDWRITE

Kurzname des JMS-Verwaltungstools: MDR

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Werte

Nein
Alle JMS_IBM_MQMD*-Eigenschaften werden ignoriert; ihre Werte werden nicht in die zugrunde liegende MQMD-Struktur kopiert. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie "False" für Programme.

YES
Die JMS_IBM_MQMD*-Eigenschaften werden verarbeitet. Ihre Werte werden in die zugrunde liegende MQMD-Struktur kopiert.

Verwenden Sie "True" für Programme.

MDMSGCTX

Welche Ebene von Nachrichtenkontext durch die JMS-Anwendung gesetzt werden soll. Die Anwendung muss mit entsprechender Kontextberechtigung ausgeführt werden, damit diese Eigenschaft in Kraft treten kann.

Gültige Objekte

Ausgeschriebener Name des JMS-Verwaltungstools: MDMSGCTX

Kurzname des JMS-Verwaltungstools: MDCTX

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Werte

DEFAULT
Im API-Aufruf MQOPEN und in der MQPMO-Struktur sind keine expliziten Optionen für den Nachrichtenkontext angegeben. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQ_MDCTX_DEFAULT".

SET_IDENTITY_CONTEXT
Der API-Aufruf MQOPEN gibt die Nachrichtenkontextoption MQOO_SET_IDENTITY_CONTEXT an und die MQPMO-Struktur gibt MQPMO_SET_IDENTITY_CONTEXT an.

Verwenden Sie für Programme "WMQ_MDCTX_SET_IDENTITY_CONTEXT".

SET_ALL_CONTEXT
Der API-Aufruf MQOPEN gibt die Nachrichtenkontextoption MQOO_SET_ALL_CONTEXT an und die MQPMO-Struktur gibt MQPMO_SET_ALL_CONTEXT an.

Verwenden Sie für Programme "WMQ_MDCTX_SET_ALL_CONTEXT".

MSGBATCHSZ

Die maximale Anzahl an Nachrichten, die bei einer asynchronen Nachrichtenzustellung aus einer Warteschlange in einem einzigen Paket abgerufen werden kann.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: MAXBUFFSIZE

Kurzname des JMS-Verwaltungstools: MBSZ

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMsgBatchSize()

- MQConnectionFactory.getMsgBatchSize()

Werte

10
Dies ist der Standardwert.

Jede beliebige positive Ganzzahl.

MSGBODY

Bestimmt, ob eine JMS-Anwendung auf den MQRFH2 einer IBM WebSphere MQ-Nachricht als Teil der Nachrichtennutzdaten zugreift.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: WMQ_MESSAGE_BODY

Kurzname des JMS-Verwaltungstools: MBODY

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Werte

UNSPECIFIED
Beim Senden von Nachrichten generiert IBM WebSphere MQ classes for JMS je nachdem, welcher Wert für WMQ_TARGET_CLIENT angegeben ist, einen MQRFH2-Header und fügt diesen ein oder auch nicht. Beim Empfang dient dieser Wert als Wert JMS.

JMS
Beim Senden von Nachrichten generiert IBM WebSphere MQ classes for JMS automatisch einen MQRFH2-Header und fügt diesen in die WebSphere MQ-Nachricht ein.

Beim Empfang von Nachrichten setzt IBM WebSphere MQ classes for JMS die JMS-Nachrichteneigenschaften entsprechend den Werten im MQRFH2-Header (falls vorhanden); der MQRFH2-Header wird nicht als Teil des JMS-Nachrichtentexts dargestellt.

MQ
Beim Senden generiert IBM WebSphere MQ classes for JMS keinen MQRFH2.

Beim Empfangen von Nachrichten stellt IBM WebSphere MQ classes for JMS den MQRFH2-Header als Teil des JMS-Nachrichtenhauptteils dar.

MSGRETENTION

Die Angabe, ob der Verbindungskonsument nicht zugestellte Nachrichten in der Eingabewarteschlange beibehält.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Ausgeschriebener Name des JMS-Verwaltungstools: MSGRETENTION

Kurzname des JMS-Verwaltungstools: MRET

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMessageRetention()
- MQConnectionFactory.getMessageRetention()

Werte

Ja
Nicht zugestellte Nachrichten bleiben in der Eingabewarteschlange. Dies ist der Standardwert.

Nein
Nicht zugestellte Nachrichten werden entsprechend ihrer Dispositionsoptionen verarbeitet.

MSGSELECTION

Bestimmt, ob eine Nachrichtenauswahl durch WebSphere MQ Classes for JMS oder durch den Broker erfolgt. Wenn TRANSPORT den WERT DIRECT aufweist, erfolgt die Nachrichtenauswahl immer über den Broker und der Wert von MSGSELECTION wird ignoriert. Eine Nachrichtenauswahl durch den Broker wird nicht unterstützt, wenn BROKERVER den Wert "V1" aufweist.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: MSGSELECTION

Kurzname des JMS-Verwaltungstools: MSEL

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMessageSelection()
- MQConnectionFactory.getMessageSelection()

Werte

CLIENT
Die Nachrichtenauswahl erfolgt durch WebSphere MQ Classes for JMS. Dies ist der Standardwert.

BROKER
Die Nachrichtenauswahl erfolgt durch den Broker.

MULTICAST

Diese Eigenschaft dient dazu, Multicasting für eine Echtzeitverbindung zu einem Broker zu aktivieren und um genau anzugeben, wie das Multicasting zum Zustellen von Nachrichten vom Broker an einen Nachrichtenkonsumenten verwendet wird. Die Eigenschaft hat keine Auswirkungen darauf, wie ein Nachrichtenproduzent Nachrichten an einen Broker sendet.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: MULTICAST

Kurzname des JMS-Verwaltungstools: MCAST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Werte

INAKTIVIERT
Die Nachrichten werden nicht mithilfe von Multicasting-Übertragung an einen Nachrichtenkonsumenten zugestellt. Dies ist der Standardwert für ConnectionFactory- und TopicConnectionFactory-Objekte.

ASCF
Die Nachrichten werden entsprechend der Multicasting-Einstellung für die Verbindungsfactory, die dem Nachrichtenkonsumenten zugeordnet ist, an den Nachrichtenkonsumenten zugestellt. Die Multicasting-Einstellung für die Verbindungsfactory wird zum Zeitpunkt des Erstellens des Nachrichtenkonsumenten berücksichtigt. Dieser Wert ist nur für Topic-Objekte gültig. Er ist der Standardwert für Topic-Objekte.

ENABLED
Wenn das Topic-Objekt für Multicasting im Broker konfiguriert ist, werden die Nachrichten mithilfe einer Multicasting-Übertragung an einen Nachrichtenkonsumenten zugestellt. Wenn das Topic-Objekt für zuverlässiges Multicasting konfiguriert ist, wird eine zuverlässige Servicequalität verwendet.

RELIABLE
Wenn das Topic-Objekt für zuverlässiges Multicasting im Broker konfiguriert ist, werden die Nachrichten mithilfe einer Multicasting-Übertragung mit zuverlässiger Servicequalität an einen Nachrichtenkonsumenten zugestellt. Ist das Topic-Objekt nicht für zuverlässiges Multicasting konfiguriert, können Sie keinen Nachrichtenkonsumenten für das Topic-Objekt erstellen.

NOTR
Wenn das Topic-Objekt für Multicasting im Broker konfiguriert ist, werden die Nachrichten mithilfe einer Multicasting-Übertragung an den Nachrichtenkonsumenten zugestellt. Auch wenn das Topic-Objekt für zuverlässiges Multicasting konfiguriert ist, wird keine zuverlässige Servicequalität verwendet.

OPTIMISTICPUBLICATION

Diese Eigenschaft bestimmt, ob WebSphere MQ Classes for JMS, unmittelbar nachdem ein Publisher eine Nachricht veröffentlicht hat, die Steuerung an diesen zurückgibt oder ob WebSphere MQ die Steuerung erst zurückgibt, nachdem alle Verarbeitungsschritte des Aufrufs abgeschlossen sind und das Ergebnis an den Publisher gemeldet werden kann.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: OPTIMISTICPUBLICATION

Kurzname des JMS-Verwaltungstools: OPTPUB

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setOptimisticPublication()
- MQConnectionFactory.getOptimisticPublication()

Werte

Nein
Wenn ein Publisher eine Nachricht veröffentlichet, gibt WebSphere MQ Classes for JMS die Steuerung erst dann an den Publisher zurück, wenn alle Verarbeitungsschritte abgeschlossen sind, die zu dem Aufruf gehören, und das Ergebnis an den Publisher gemeldet werden kann. Dies ist der Standardwert.

YES
Wenn ein Publisher eine Nachricht veröffentlichet, gibt WebSphere MQ Classes for JMS die Steuerung sofort an den Publisher zurück, bevor alle Verarbeitungsschritte abgeschlossen sind, die zu dem Aufruf gehören, und das Ergebnis an den Publisher gemeldet werden kann. WebSphere MQ Classes for JMS meldet das Ergebnis nur, wenn der Publisher die Nachricht festschreibt.

OUTCOMENOTIFICATION

Diese Eigenschaft bestimmt, ob WebSphere MQ Classes for JMS die Steuerung sofort an einen Subskribenten zurückgibt, der eine Nachricht bestätigt oder festgeschrieben hat, oder ob WebSphere MQ die Steuerung erst zurückgibt, nachdem alle Verarbeitungsschritte des Aufrufs abgeschlossen sind und das Ergebnis an den Subskribenten gemeldet werden kann.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: OUTCOMENOTIFICATION

Kurzname des JMS-Verwaltungstools: NOTIFY

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setOutcomeNotification()
- MQConnectionFactory.getOutcomeNotification()

Werte

YES
Wenn ein Subskribent eine Nachricht bestätigt oder festschreibt, gibt WebSphere MQ Classes for JMS die Steuerung erst dann an den Subskribenten zurück, wenn alle Verarbeitungsschritte abgeschlossen sind, die zu dem Aufruf gehören, und das Ergebnis an den Subskribenten gemeldet werden kann. Dies ist der Standardwert.

Nein
Wenn ein Subskribent eine Nachricht bestätigt oder festschreibt, gibt WebSphere MQ Classes for JMS die Steuerung sofort an den Subskribenten zurück, bevor alle Verarbeitungsschritte abgeschlossen sind, die zu dem Aufruf gehören, und das Ergebnis an den Subskribenten gemeldet werden kann.

PERSISTENCE

Die Permanenz von Nachrichten, die an eine Zieladresse gesendet werden.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: PERSISTENCE

Kurzname des JMS-Verwaltungstools: PER

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Werte

APP
Die Persistenz wird durch die JMS-Anwendung definiert. Dies ist der Standardwert.

QDEF
Die Persistenz übernimmt den Wert des für die Warteschlange definierten Standardwerts.

PERS
Nachrichten sind persistent.

NON
Nachrichten sind nicht persistent.

HIGH
Informationen zur Verwendung dieses Werts finden Sie im Abschnitt Persistente JMS-Nachrichten.

POLLINGINT

Wenn sich bei den einzelnen Nachrichtenlistenern innerhalb einer Sitzung keine geeignete Nachricht in der zugehörigen Warteschlange befindet, ist dies das maximale Intervall in Millisekunden, das verstreicht, bevor die einzelnen Nachrichtenlistener erneut versuchen, eine Nachricht aus der zugehörigen Warteschlange abzurufen. Wenn regelmäßig keine geeigneten Nachrichten für die Nachrichtenlistener innerhalb einer Sitzung verfügbar sind, sollten Sie einen höheren Wert für diese Eigenschaft angeben. Diese Eigenschaft ist nur relevant, wenn für TRANSPORT der Wert BIND oder CLIENT festgelegt ist.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: POLLINGINT

Kurzname des JMS-Verwaltungstools: PINT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setPollingInterval()
- MQConnectionFactory.getPollingInterval()

Werte

5000
Dies ist der Standardwert.

Jede beliebige positive Ganzzahl.

PORT

Für eine Verbindung zu einem Warteschlangenmanager die Nummer des Ports, für den der Warteschlangenmanager empfangsbereit ist, oder für eine Echtzeitverbindung zu einem Broker die Nummer des Ports, für den der Broker für Echtzeitverbindungen empfangsbereit ist.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-ConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PORT

Kurzname des JMS-Verwaltungstools: PORT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

Werte

1414
Dies ist der Standardwert, wenn für TRANSPORT der Wert CLIENT festgelegt ist.

1506
Dies ist der Standardwert, wenn für TRANSPORT der Wert DIRECT oder DIRECTHTTP festgelegt ist.

Jede beliebige positive Ganzzahl.

PRIORITY

Die Priorität von Nachrichten, die an eine Zieladresse gesendet werden.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: PRIORITY

Kurzname des JMS-Verwaltungstools: PRI

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setPriority()
- MQDestination.getPriority()

Werte

APP
Die Priorität wird durch die JMS-Anwendung definiert. Dies ist der Standardwert.

QDEF
Die Priorität übernimmt den Wert des für die Warteschlange definierten Standardwerts.

Jede beliebige Ganzzahl im Bereich 0 bis 9.
Von der niedrigsten zur höchsten.

PROCESSDURATION

Diese Eigenschaft bestimmt, ob ein Subskribent zusichert, jede empfangene Nachricht schnell zu verarbeiten, bevor er die Steuerung an WebSphere MQ Classes for JMS zurückgibt.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PROCESSDURATION

Kurzname des JMS-Verwaltungstools: PROCDUR

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setProcessDuration()
- MQConnectionFactory.getProcessDuration()

Werte

UNBEKANNT
Ein Subskribent kann keine Aussage darüber machen, wie schnell empfangene Nachrichten verarbeitet werden. Dies ist der Standardwert.

SHORT
Ein Subskribent sichert zu, jede empfangene Nachricht schnell zu verarbeiten, bevor er die Steuerung an WebSphere MQ Classes for JMS zurückgibt.

PROVIDERVERSION

Diese Eigenschaft unterscheidet zwischen den zwei Messaging-Betriebsarten von WebSphere MQ: den WebSphere MQ-Normalmodus für Messaging-Provider und den WebSphere MQ-Migrationsmodus für Messaging-Provider.

Der WebSphere MQ-Normalmodus für Messaging-Provider verwendet alle Features der WebSphere MQ Version 7.0 Warteschlangenmanager zur Implementierung von JMS. Dieser Modus wird nur zur Verbindung mit einem WebSphere MQ Warteschlangenmanager verwendet und kann zur Verbindung mit WebSphere MQ Version 7.0 Warteschlangenmanagern entweder in Client- oder Bindungsmodus verwendet werden. Dieser Modus ist optimiert für die Verwendung mit der neuen WebSphere MQ Version 7.0-Funktion. Wenn Sie WebSphere MQ-Echtzeittransport nicht verwenden, wird die Betriebsart vorrangig durch die Eigenschaft PROVIDERVERSION der Verbindungsfactory bestimmt.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueue-Connection Factory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PROVIDERVERSION

Kurzname des JMS-Verwaltungstools: PVER

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setProviderVersion()
- MQConnectionFactory.getProviderVersion()

Werte

Sie können **PROVIDERVERSION** auf die möglichen Werte 7, 6 oder *unspecified* setzen. **PROVIDERVERSION** kann jedoch eine Zeichenfolge in einem der folgenden Formate sein:

- V.R.M.F
- V.R.M
- V.R
- V

Dabei sind V, R, M und F Ganzzahlen größer oder gleich Null.

7

Verwendet den WebSphere MQ-Normalmodus für Messaging-Provider.

Wenn PROVIDERVERSION auf 7 gesetzt wird, steht nur die Betriebsart WebSphere MQ-Normalmodus für Messaging-Anbieter zur Verfügung. Wenn es sich bei dem aufgrund der anderen Einstellungen in der Verbindungsfactory verbundenen Warteschlangenmanager nicht um einen Version 7.0-Warteschlangenmanager handelt, schlägt die Methode "createConnection()" mit einer Ausnahmebedingung fehl.

Der WebSphere MQ-Normalmodus für Messaging-Provider verwendet die Komponente zur gemeinsamen Nutzung von Dialogen. Die Anzahl der gemeinsam genutzten Dialoge wird von der Eigenschaft "SHARECNV()" auf dem Serververbindungskanal gesteuert. Wenn für diese Eigenschaft der Wert "0" festgelegt ist, können Sie den WebSphere MQ-Normalmodus für Messaging-Provider nicht verwenden und die Methode "createConnection()" schlägt mit einer Ausnahmebedingung fehl.

6

Verwendet den WebSphere MQ-Migrationsmodus für Messaging-Provider.

WebSphere MQ Classes for JMS verwendet die in WebSphere MQ Version 6.0 enthaltenen Komponenten und Algorithmen. Wenn Sie mit dem WebSphere Event Broker oder dem WebSphere Message Broker unter Verwendung von WebSphere MQ Enterprise Transport eine Verbindung herstellen möchten, müssen Sie diesen Modus verwenden. Mit diesem Modus können Sie zwar eine Verbindung mit einem WebSphere MQ Version 7.0 Warteschlangenmanager herstellen, es werden jedoch keine der neuen Features der Version 7.0 Warteschlangenmanager, wie zum Beispiel Vorauslesen oder Streaming verwendet.

nicht angegeben

Dies ist der Standardwert und der Text lautet "unspecified".

Eine mit einer vorherigen Version von WebSphere MQ Classes for JMS in JNDI erstellte Verbindungsfactory hat diesen Wert, wenn sie mit der neuen Version von WebSphere MQ Classes for JMS verwendet wird. Der folgende Algorithmus wird zur Bestimmung der Betriebsart verwendet. Dieser Algorithmus wird verwendet, wenn die Methode "createConnection()" aufgerufen wird und andere Aspekte der Verbindungsfactory verwendet werden, um zu bestimmen, ob der WebSphere MQ-Normalmodus für Messaging-Provider oder der WebSphere MQ-Migrationsmodus für Messaging-Provider erforderlich ist.

- Es wird zuerst versucht, den WebSphere MQ-Normalmodus für Messaging-Provider zu verwenden.
- Falls der verbundene Warteschlangenmanager kein WebSphere MQ Version 7.0 Warteschlangenmanager ist, wird die Verbindung geschlossen und der WebSphere MQ-Migrationsmodus für Messaging-Provider verwendet.
- Wenn für die Eigenschaft SHARECNV() auf dem Serververbindungskanal der Wert "0" festgelegt ist, wird die Verbindung geschlossen und der WebSphere MQ-Migrationsmodus für Messaging-Provider verwendet.
- Wenn für BROKERVER der Wert "1" oder der neue Standardwert "unspecified" festgelegt ist, wird weiterhin der WebSphere MQ-Normalmodus für Messaging-Provider verwendet, weshalb alle Publish/Subscribe-Operationen die neuen Features von WebSphere MQ Version 7.0 verwenden. Wenn der WebSphere Event Broker oder WebSphere Message Broker im Kompatibilitätsmodus verwendet wird (und Sie statt der Publish/Subscribe-Funktion von WebSphere MQ Version 7 die Publish/Subscribe-Funktion von Version 6.0 verwenden möchten), legen Sie für PROVIDERVERSION den Wert "6" fest, um sicherzustellen, dass der WebSphere MQ-Migrationsmodus für den Messaging-Provider verwendet wird.

PROXYHOSTNAME

Der Hostname oder die IP-Adresse des Systems, auf dem der Proxy-Server ausgeführt wird, wenn eine Echtzeitverbindung zu einem Broker über einen Proxy-Server verwendet wird.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PROXYHOSTNAME

Kurzname des JMS-Verwaltungstools: PHOST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Werte

null
Den Hostnamen des Proxy-Servers. Dies ist der Standardwert.

PROXYPORT

Die Nummer des Ports, für den der Proxy-Server empfangsbereit ist, wenn eine Echtzeitverbindung zu einem Broker über einen Proxy-Server verwendet wird.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PROXYPORT

Kurzname des JMS-Verwaltungstools: PPORT

Programmgesteuerter Zugriff

Setter/Getter

MQConnectionFactory.setProxyPort()

MQConnectionFactory.getProxyPort()

Werte

443
Die Portnummer des Proxy-Servers. Dies ist der Standardwert.

PUBACKINT

Die Anzahl an Nachrichten, die von einem Publisher veröffentlicht werden, bevor WebSphere MQ Classes for JMS eine Bestätigung vom Broker anfordert.

Wenn Sie den Wert für diese Eigenschaft herabsetzen, fordert WebSphere MQ Classes for JMS häufiger eine Bestätigung an, was zu einer schlechteren Leistung des Publishers führt. Wenn Sie den Wert erhöhen, benötigt WebSphere MQ Classes for JMS mehr Zeit, um eine Ausnahmebedingung auslösen, falls der Broker fehlschlägt. Diese Eigenschaft ist nur relevant, wenn für TRANSPORT der Wert BIND oder CLIENT festgelegt ist.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: PROXYPORT

Kurzname des JMS-Verwaltungstools: PPORT

Programmgesteuerter Zugriff

Setter/Getter

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Werte

25
Jede beliebige positive Ganzzahl kann der Standardwert sein.

PUTASYNCALLOWED

Diese Eigenschaft gibt an, ob Nachrichtenproduzenten asynchrone PUT-Operationen verwenden dürfen, um Nachrichten an diese Zieladresse zu senden.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: PUTASYNCALLOWED

Kurzname des JMS-Verwaltungstools: PAALD

Programmgesteuerter Zugriff

Setter/Getter

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Werte

AS_DEST
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Queue- oder des Topic-Objekts verwiesen wird. Dies ist der Standardwert.

AS_Q_DEF
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Queue-Objekts verwiesen wird.

AS_TOPIC_DEF
Bestimmt, ob asynchrone PUT-Operationen zulässig sind, indem auf die Definition des Topic-Objekts verwiesen wird.

Nein
Asynchrone PUT-Operationen sind nicht zulässig.

YES
Asynchrone PUT-Operationen sind zulässig.

QMANAGER

Der Name des Warteschlangenmanagers, zu dem eine Verbindung hergestellt werden soll.

Wenn Ihre Anwendung eine Definitionstabelle für Clientkanäle zum Herstellen einer Verbindung zu einem Warteschlangenmanager verwendet, sollten Sie die Informationen im Abschnitt Definitionstabelle für Clientkanäle mit WebSphere MQ Classes for JMS verwenden lesen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XA-QueueConnection Factory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: QMANAGER

Kurzname des JMS-Verwaltungstools: QMGR

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setQueueManager()
- MQConnectionFactory.getQueueManager()

Werte

"" (leere Zeichenfolge)
Jede beliebige Zeichenfolge kann der Standardwert sein.

WARTESCHLANGE

Gibt den Namen des JMS-Warteschlangenziels an. Dieser stimmt mit den Namen der Warteschlange überein, die vom Warteschlangenmanager verwendet wird.

Gültige Objekte

Warteschlange

Ausgeschriebener Name des JMS-Verwaltungstools: QUEUE

Kurzname des JMS-Verwaltungstools: QU

Werte

Jede beliebige Zeichenfolge.
Jeder gültige IBM WebSphere MQ-Warteschlangenname.

Zugehörige Konzepte
Regeln für die Benennung von IBM WebSphere MQ-Objekten

READAHEADALLOWED

Diese Eigenschaft gibt an, ob es zulässig ist, dass Nachrichtenkonsumenten und Warteschlangenbrowser die Vorauslesefunktion verwenden, um nicht permanente Nachrichten von dieser Zieladresse in einen internen Puffer abzurufen, bevor sie sie empfangen.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: READAHEADALLOWED

Kurzname des JMS-Verwaltungstools: RAALD

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Werte

AS_DEST
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Queue- oder des Topic-Objekts verwiesen wird. Dies ist der Standardwert in Verwaltungstools.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST".

AS_Q_DEF
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Queue-Objekts verwiesen wird.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF".

AS_TOPIC_DEF
Bestimmt, ob die Vorauslesefunktion zulässig ist, indem auf die Definition des Topic-Objekts verwiesen wird.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF".

Nein
Die Vorauslesefunktion ist nicht zulässig.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED".

YES
Die Vorauslesefunktion ist zulässig.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED".

READAHEADCLOSEPOLICY

Diese Eigenschaft gibt für Nachrichten, die an einen asynchronen Nachrichtenlistener zugestellt werden, an, was mit Nachrichten im internen Vorauslesepuffer geschehen soll, wenn der Nachrichtenkonsument geschlossen wird.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: READAHEADCLOSEPOLICY

Kurzname des JMS-Verwaltungstools: RACP

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Werte

DELIVER_ALL
Alle Nachrichten im internen Vorauslesepuffer werden vor der Rückgabe an den Nachrichtenlistener der Anwendung zugestellt. Dies ist der Standardwert in Verwaltungstools.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_DELIVERALL".

DELIVER_CURRENT
Vor der Rückgabe wird nur der aktuelle Nachrichtenlisteneraufruf abgeschlossen, wobei Nachrichten im internen Vorauslesepuffer verbleiben können, die anschließend gelöscht werden.

Verwenden Sie in Programmen "WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT".

RECEIVECCSID

Die Zieleigenschaft, die die Ziel-CCSID für die Nachrichtenkonvertierung des Warteschlangenmanagers festlegt. Der Wert wird ignoriert, es sei denn, für RECEIVECONVERSION ist der Wert `WMQ_RECEIVE_CONVERSION_QMGR` festgelegt.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: RECEIVECCSID

Kurzname des JMS-Verwaltungstools: RCCS

Programmgesteuerter Zugriff

Setter/Getter

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Werte

`WMQConstants.WMQ_RECEIVE_CCSID_JVM_DEFAULT`
0 - `Charset.defaultCharset` der JVM verwenden

1208
UTF-8

CCSID
Die ID des unterstützten codierten Zeichensatzes.

RECEIVECONVERSION

Die Zieleigenschaft, die bestimmt, ob eine Datenkonvertierung vom Warteschlangenmanager durchgeführt wird.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: RECEIVECONVERSION

Kurzname des JMS-Verwaltungstools: RCNV

Programmgesteuerter Zugriff

Setter/Getter

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Werte

`WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG`
1 - Die Datenkonvertierung wird nur auf dem JMS-Client durchgeführt. Dies ist der Standardwert bis Version 7.0 und ab Version 7.0.1.5 (einschließlich).

`WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR`
2 - Im Warteschlangenmanager wird eine Datenkonvertierung durchgeführt, bevor eine Nachricht an den Client gesendet wird. Dies ist der (einzig mögliche) Standardwert ab Version 7.0 bis Version 7.0.1.4 einschließlich, außer wenn APAR IC72897 angewendet wurde.

RECEIVEISOLATION

Diese Eigenschaft bestimmt, ob ein Subskribent Nachrichten empfangen kann, die in der Warteschlange für Subskribenten nicht festgeschrieben wurden.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: RECEIVEISOLATION

Kurzname des JMS-Verwaltungstools: RCVISOL

Werte

COMMITTED
Ein Subskribent empfängt nur die Nachrichten in der Warteschlange für Subskribenten, die festgeschrieben wurden. Dies ist der Standardwert in Verwaltungstools.

Verwenden Sie in Programmen "WMQConstants.WMQ_RCVISOL_COMMITTED".

UNCOMMITTED
Ein Subskribent kann Nachrichten empfangen, die in der Warteschlange für Subskribenten nicht festgeschrieben wurden.

Verwenden Sie in Programmen "WMQConstants.WMQ_RCVISOL_UNCOMMITTED".

RECEXIT

Gibt einen Kanalempfangsexit oder eine Folge von Empfangsexits an, die in einer bestimmten Reihenfolge ausgeführt werden sollen.

Es ist möglicherweise eine zusätzliche Konfiguration erforderlich, damit die IBM WebSphere MQ classes for JMS Empfangsexits empfangen können. Weitere Informationen finden Sie im Abschnitt IBM WebSphere MQ-Klassen für JMS zur Verwendung von Kanalexits konfigurieren.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: RECEXIT

Kurzname des JMS-Verwaltungstools: RCX

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setReceiveExit()
- MQConnectionFactory.getReceiveExit()

Werte

null
Eine Zeichenfolge, die aus einem oder mehr Elementen besteht, die durch Kommata voneinander getrennt sind, wobei jedes dieser Elemente eins der beiden folgenden ist:

- Der Name einer Klasse, die die Schnittstelle WMQReceiveExit (für einen in Java geschriebenen Kanalempfangsexit) implementiert.
- Eine Zeichenfolge im Format *libraryName(entryPointName)* (für einen Kanalempfangsexit, der nicht in Java geschrieben ist).

Dies ist der Standardwert.

RECEXITINIT

Die Benutzerdaten, die beim Aufruf von Kanalempfangsexits an diese übergeben werden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: RECEXITINIT

Kurzname des JMS-Verwaltungstools: RCXI

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Werte

null
Eine Zeichenfolge, die aus einem oder mehr Benutzerdatenelementen besteht, die voneinander durch Kommata getrennt sind. Dies ist der Standardwert.

REPLYTOSTYLE

Bestimmt, wie das Feld JMSReplyTo in einer erhaltenen Nachricht erstellt wird.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: REPLYTOSTYLE

Kurzname des JMS-Verwaltungstools: RTOST

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Werte

DEFAULT
Äquivalent zu MQMD.

RFH2
Es wird der Wert verwendet, der im RFH2-Header angegeben wurde. Wenn ein JMSReplyTo-Wert in der sendenden Anwendung gesetzt worden ist, wird dieser Wert verwendet.

MQMD
Der von MQMD angegebene Wert wird verwendet. Dieses Verhalten ist äquivalent zum Standardverhalten von WebSphere MQ Version 6.0.2.4 und 6.0.2.5.

Wenn der JMSReplyTo-Wert, der von der sendenden Anwendung gesetzt wurde, keinen Warteschlangenmanagernamen enthält, fügt der empfangende Warteschlangenmanager seinen eigenen Namen in MQMD ein. Wenn Sie diesen Parameter auf MQMD setzen, befindet sich die verwendete Warteschlange für Antwortnachrichten beim empfangenden Warteschlangenmanager. Wenn Sie diesen Parameter auf

RFH2 setzen, befindet sich die verwendete Warteschlange für Antwortnachrichten bei dem Warteschlangenmanager, der im RFH2 der gesendeten Nachricht angegeben ist, wie ursprünglich von der sendenden Anwendung gesetzt.

Wenn der JMSReplyTo-Wert, der von der sendenden Anwendung gesetzt wurde, einen Warteschlangenmanagernamen enthält, ist der Wert dieses Parameters nicht relevant, da sowohl MQMD als auch RFH2 denselben Wert enthalten.

RESCANINT

Wenn ein Nachrichtenkonsument in der Punkt-zu-Punkt-Domäne mithilfe eines Nachrichtenselektors die Nachrichten auswählt, die er empfangen möchte, durchsucht WebSphere MQ Classes for JMS die WebSphere MQ-Warteschlange nach geeigneten Nachrichten in der Reihenfolge, die durch das Attribut `MsgDeliverySequence` der Warteschlange festgelegt ist.

Nachdem WebSphere MQ Classes for JMS eine geeignete Nachricht gefunden und sie an den Konsumenten zugestellt hat, nimmt WebSphere MQ Classes for JMS die Suche nach der nächsten geeigneten Nachricht von der aktuellen Position in der Warteschlange ausgehend wieder auf. WebSphere MQ Classes for JMS setzt die Suche in der Warteschlange auf diese Weise fort, bis das Ende der Warteschlange erreicht ist oder das über diese Eigenschaft in Millisekunden festgelegte Intervall abläuft. In beiden Fällen kehrt WebSphere MQ Classes for JMS an den Anfang der Warteschlange zurück, um die Suche fortzusetzen, und ein neues Intervall beginnt.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: RESCANINT

Kurzname des JMS-Verwaltungstools: RINT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setRescanInterval()
- MQConnectionFactory.getRescanInterval()

Werte

5000
Jede beliebige positive Ganzzahl kann der Standardwert sein.

SECEXIT

Gibt einen Kanalsicherheitsexit an.

Es ist möglicherweise eine zusätzliche Konfiguration erforderlich, damit die IBM WebSphere MQ classes for JMS Sicherheitsexits empfangen können. Weitere Informationen finden Sie im Abschnitt IBM WebSphere MQ-Klassen für JMS zur Verwendung von Kanalexits konfigurieren.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SECEXIT

Kurzname des JMS-Verwaltungstools: SXC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSecurityExit()
- MQConnectionFactory.getSecurityExit()

Werte

null

Der Name einer Klasse, die die Schnittstelle WMQSecurityExit implementiert (für einen in Java geschriebenen Kanalsicherheitsexit).

Eine Zeichenfolge im Format *libraryName(entryPointName)* (für einen Kanalsicherheitsexit, der nicht in Java geschrieben ist).

SECEXITINIT

Die Benutzerdaten, die beim Aufruf des Kanalsicherheitsexits an diesen übergeben werden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SECEXITINIT

Kurzname des JMS-Verwaltungstools: SCXI

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Werte

null

Jede beliebige Zeichenfolge kann der Standardwert sein.

SENDCHECKCOUNT

Die Anzahl der Sendeaufrufe, die zwischen den Prüfungen auf Fehler innerhalb einer einzelnen JMS-Sitzung ohne Transaktion bei der asynchronen PUT-Operation zugelassen werden sollen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SENDCHECKCOUNT

Kurzname des JMS-Verwaltungstools: SCC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Werte

null
Jede beliebige Zeichenfolge kann der Standardwert sein.

SENDEXIT

Gibt einen Kanalsendesexit oder eine Folge von Sendeexits an, die in einer bestimmten Reihenfolge ausgeführt werden sollen.

Es ist möglicherweise eine zusätzliche Konfiguration erforderlich, damit die IBM WebSphere MQ classes for JMS Sendeexits empfangen können. Weitere Informationen finden Sie im Abschnitt IBM WebSphere MQ-Klassen für JMS zur Verwendung von Kanalexits konfigurieren.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SENDEXIT

Kurzname des JMS-Verwaltungstools: SDX

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSendExit()
- MQConnectionFactory.getSendExit()

Werte

null
Jede beliebige Zeichenfolge, die aus einem oder mehr Elementen besteht, die durch Kommata voneinander getrennt sind, wobei jedes dieser Elemente eins der beiden folgenden ist:

- Der Name einer Klasse, die die Schnittstelle WMQSendExit (für einen in Java geschriebenen Kanalsendeexit) implementiert.
- Eine Zeichenfolge im Format *libraryName(entryPointName)* (für einen Kanalsendeexit, der nicht in Java geschrieben ist)
-

Dies ist der Standardwert.

SENDEXITINIT

Die Benutzerdaten, die beim Aufruf von Kanalsendeexits an diese übergeben werden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SENDEXITINIT

Kurzname des JMS-Verwaltungstools: SDXI

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSendExitInit()

- MQConnectionFactory.getSendExitInit()

Werte

null
Jede beliebige Zeichenfolge, die aus einem oder mehr Benutzerdatenelementen besteht, die voneinander durch Kommata getrennt sind, kann der Standardwert sein.

SHARECONVALLOWED

Diese Eigenschaft bestimmt, ob eine Clientverbindung ihren Socket gemeinsam mit anderen JMS-Verbindungen der obersten Ebene vom gleichen Prozess zum gleichen Warteschlangenmanager nutzen kann, wenn die Kanaldefinitionen übereinstimmen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SHARECONVALLOWED

Kurzname des JMS-Verwaltungstools: SCALD

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Werte

YES
Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES".

Nein
Dieser Wert ist für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO".

SPARSESUBS

Diese Eigenschaft steuert die Richtlinie für den Nachrichtenabruf eines TopicSubscriber-Objekts.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SPARSESUBS

Kurzname des JMS-Verwaltungstools: SSUBS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSparseSubscriptions()
- MQConnectionFactory.getSparseSubscriptions()

Werte

Nein
Die Subskriptionen empfangen regelmäßige Abgleichnachrichten. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie "false" für Programme.

YES
Die Subskriptionen empfangen unregelmäßige Abgleichnachrichten. Für diesen Wert ist es erforderlich, dass die Subskriptionswarteschlange zum Durchsuchen geöffnet werden kann.

Verwenden Sie "true" für Programme.

SSLCIPHERSUITE

Die Cipher-Suite, die für eine SSL-Verbindung verwendet werden soll.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SSLCIPHERSUITE

Kurzname des JMS-Verwaltungstools: SCPHS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSSLCipherSuite()
- MQConnectionFactory.getSSLCipherSuite()

Werte

null
Dies ist der Standardwert. Weitere Informationen finden Sie im Abschnitt SSL-Eigenschaften von JMS-Objekten.

SSLCRL

Server mit Zertifikatswiderruflisten, die auf SSL-Zertifikatswiderrufe geprüft werden sollen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SSLCRL

Kurzname des JMS-Verwaltungstools: SCRL

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSSLCertStores()
- MQConnectionFactory.getSSLCertStores()

Werte

null
Eine Liste mit LDAP-URLs, die durch Leerzeichen voneinander getrennt sind. Dies ist der Standardwert. Weitere Informationen finden Sie im Abschnitt SSL-Eigenschaften von JMS-Objekten.

SSLFIPSREQUIRED

Diese Eigenschaft bestimmt, ob eine SSL-Verbindung eine CipherSuite verwenden muss, die vom IBM Java-JSSE-FIPS-Provider (IBMJSSEFIPS) unterstützt wird.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SSLFIPSREQUIRED

Kurzname des JMS-Verwaltungstools: SFIPS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSSLFipsRequired()
- MQConnectionFactory.getSSLFipsRequired()

Werte

Nein
Eine SSL-Verbindung kann jede CipherSuite verwenden, die nicht vom IBM Java JSSE FIPS-Provider (IBMJSSEFIPS) unterstützt wird.

Dies ist der Standardwert. Verwenden Sie in Programmen "false".

YES
Eine SSL-Verbindung muss eine CipherSuite verwenden, die von IBMJSSEFIPS unterstützt wird.

Verwenden Sie in Programmen "true".

SSLPEERNAME

Unter SSL ein Entwurf für einen *definierten Namen*, der mit dem durch den Warteschlangenmanager bereitgestellten Namen übereinstimmen muss.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SSLPEERNAME

Kurzname des JMS-Verwaltungstools: SPEER

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSSLPeerName()
- MQConnectionFactory.getSSLPeerName()

Werte

null
Dies ist der Standardwert. Weitere Informationen finden Sie im Abschnitt SSL-Eigenschaften von JMS-Objekten.

SSLRESETCOUNT

Die Gesamtzahl der von einer Verbindung gesendeten und empfangenen Bytes, bevor der geheime Schlüssel für die Verschlüsselung erneut vereinbart wird.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SSLRESETCOUNT

Kurzname des JMS-Verwaltungstools: SRC

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSSLResetCount()
- MQConnectionFactory.getSSLResetCount()

Werte

0
Null oder eine beliebige positive Ganzzahl, die kleiner-gleich 999.999.999 ist. Dies ist der Standardwert. Weitere Informationen finden Sie im Abschnitt SSL-Eigenschaften von JMS-Objekten.

STATREFRESHINT

Das Aktualisierungsintervall der lange aktiven Transaktion (in Millisekunden), die feststellt, wenn ein Subskribent die Verbindung zum Warteschlangenmanager verliert.

Diese Eigenschaft ist nur relevant, wenn für SUBSTORE der Wert QUEUE festgelegt ist.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: STATREFRESHINT

Kurzname des JMS-Verwaltungstools: SRI

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Werte

6000
Jede beliebige positive Ganzzahl kann der Standardwert sein. Weitere Informationen finden Sie im Abschnitt SSL-Eigenschaften von JMS-Objekten.

SUBSTORE

Diese Eigenschaft gibt an, wo WebSphere MQ Classes for JMS persistente Daten speichert, die in Beziehung zu aktiven Subskriptionen stehen.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SUBSTORE

Kurzname des JMS-Verwaltungstools: SS

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSubscriptionStore()
- MQConnectionFactory.getSubscriptionStore()

Werte

BROKER
Der brokerbasierte Subskriptionsspeicher wird zum Speichern von Subskriptionsdetails verwendet. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_SUBSTORE_BROKER".

MIGRATE
Überträgt Subskriptionsdaten aus dem warteschlangenbasierten Subskriptionsspeicher in den brokerbasierten Subskriptionsspeicher.

Verwenden Sie für Programme "WMQConstants.WMQ_SUBSTORE_MIGRATE".

WARTESCHLANGE
Der warteschlangenbasierte Subskriptionsspeicher wird zum Speichern von Subskriptionsdetails verwendet.

Verwenden Sie für Programme "WMQConstants.WMQ_SUBSTORE_QUEUE".

SYNCPOINTALLGETS

Diese Eigenschaft bestimmt, ob alle Get-Methoden mit Synchronisationspunkt ausgeführt werden sollen.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: SYNCPOINTALLGETS

Kurzname des JMS-Verwaltungstools: SPAG

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Werte

Nein
Dies ist der Standardwert.

Ja

TARGCLIENT

Diese Eigenschaft bestimmt, ob das WebSphere MQ-RFH2-Format zum Austauschen von Daten mit den Zielanwendungen verwendet wird.

Gültige Objekte

Queue, Topic

Ausgeschriebener Name des JMS-Verwaltungstools: TARGCLIENT

Kurzname des JMS-Verwaltungstools: TC

Programmgesteuerter Zugriff

Setter/Getter

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Werte

JMS
Das Ziel der Nachricht ist eine JMS-Anwendung. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_CLIENT_JMS_COMPLIANT".

MQ
Das Ziel der Nachricht ist eine Nicht-JMS-WebSphere MQ-Anwendung.

Verwenden Sie für Programme "WMQConstants.WMQ_CLIENT_NONJMS_MQ".

TARGCLIENTMATCHING

Diese Eigenschaft gibt an, ob eine Antwortnachricht, die an die Warteschlange gesendet wird, die durch das Headerfeld "JMSReplyTo" einer eingehenden Nachricht identifiziert wird, nur dann einen MQRFH2-Header enthält, wenn die eingehende Nachricht einen MQRFH2-Header enthält.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: TARGCLIENTMATCHING

Kurzname des JMS-Verwaltungstools: TCM

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Werte

YES
Verfügt eine eingehende Nachricht nicht über einen MQRFH2-Header, wird die Eigenschaft TARGCLIENT des Queue-Objekts, das vom Headerfeld "JMSReplyTo" der Nachricht abgeleitet wird, an MQ gesendet. Verfügt die Nachricht über einen MQRFH2-Header, wird für die Eigenschaft TARGCLIENT stattdessen der Wert "JMS" festgelegt. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie "true" für Programme.

Nein
Für die Eigenschaft TARGCLIENT des Queue-Objekts, das vom Headerfeld "JMSReplyTo" einer eingehenden Nachricht abgeleitet wird, wird immer der Wert "JMS" festgelegt.

Verwenden Sie "false" für Programme.

TEMPMODEL

Der Name der Modellwarteschlange, anhand der temporäre JMS-Warteschlangen erstellt werden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: TEMPMODEL

Kurzname des JMS-Verwaltungstools: TM

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setTemporaryModel()
- MQConnectionFactory.getTemporaryModel()

Werte

SYSTEM.DEFAULT.MODEL.QUEUE
Jede beliebige Zeichenfolge kann der Standardwert sein.

TEMPQPREFIX

Das Präfix, das verwendet wird, um den Namen einer dynamischen WebSphere MQ-Warteschlange zu bilden.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: TEMPQPREFIX

Kurzname des JMS-Verwaltungstools: TQP

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setTempQPrefix()
- MQConnectionFactory.getTempQPrefix()

Werte

" " (leere Zeichenfolge)
Als Präfix wird `CSQ.*` unter z/OS und `AMQ.*` auf allen anderen Plattformen verwendet. Dies sind die Standardwerte.

Warteschlangenpräfix
Das Warteschlangenpräfix ist eine beliebige Zeichenfolge, die den Regeln zur Bildung des Inhalts des Felds *DynamicQName* in einem WebSphere MQ-Objektdeskriptor (Struktur MQOD) entspricht. Das letzte Zeichen, bei dem es sich nicht um ein Leerzeichen handelt, muss jedoch ein Stern (*) sein.

TEMPTOPICPREFIX

Beim Erstellen temporärer Themen generiert JMS eine Themenzeichenfolge im Format ''TEMP/*TEMPTOPICPREFIX/eindeutige_ID*'' oder im Format ''TEMP/*eindeutige_ID*'', wenn diese Eigenschaft ihren Standardwert beibehält. Durch Angabe eines Wertes für die Eigenschaft TEMPTOPICPREFIX können Sie spezifische Modellwarteschlangen zum Erstellen der verwalteten Warteschlangen für Subskribenten von temporären Themen unter Verwendung der jeweiligen Verbindung definieren.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: TEMPTOPICPREFIX

Kurzname des JMS-Verwaltungstools: TTP

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Werte

Jede beliebige Zeichenfolge, die nicht null ist und die nur aus gültigen Zeichen für eine WebSphere MQ-Themenzeichenfolge besteht. Der Standartwert ist '' '' (leere Zeichenfolge).

TOPIC

Dieser Wert ist der Name des JMS-Themaziels, der vom Warteschlangenmanager als Themenzeichenfolge einer Veröffentlichung oder Subskription verwendet wird.

Gültige Objekte

Thema

Ausgeschriebener Name des JMS-Verwaltungstools: TOPIC

Kurzname des JMS-Verwaltungstools: TOP

Werte

Jede beliebige Zeichenfolge.
Eine Zeichenfolge, die eine gültige IBM WebSphere MQ-Themenzeichenfolge bildet. Bei Verwendung von IBM WebSphere MQ als Messaging-Provider für WebSphere Application Server müssen Sie als Wert den Namen angeben, unter dem das Thema innerhalb der Verwaltung von WebSphere Application Server bekannt ist.

Zugehörige Konzepte
Themenzeichenfolgen

TRANSPORT

Die Art der Verbindung zu einem Warteschlangenmanager oder Broker.

Gültige Objekte

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: TRANSPORT

Kurzname des JMS-Verwaltungstools: TRAN

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setTransportType()
- MQConnectionFactory.getTransportType()

Werte

BIND
Für eine Verbindung zu einem Warteschlangenmanager im Bindungsmodus. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_CM_BINDINGS".

CLIENT
Für eine Verbindung zu einem Warteschlangenmanager im Clientmodus.

Verwenden Sie für Programme "WMQConstants.WMQ_CM_CLIENT".

DIRECT
Für eine Echtzeitverbindung zu einem Broker ohne HTTP-Tunnelungsverfahren.

Verwenden Sie für Programme "WMQConstants.WMQ_CM_DIRECT_TCPIP".

DIRECTHTTP
Für eine Echtzeitverbindung zu einem Broker mit HTTP-Tunnelungsverfahren. Es wird nur HTTP 1.0 unterstützt.

Verwenden Sie für Programme "WMQConstants.WMQ_CM_DIRECT_HTTP".

WILDCARDFORMAT

Diese Eigenschaft gibt an, welche Version der Platzhaltersyntax verwendet werden soll.

Gültige Objekte

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Ausgeschriebener Name des JMS-Verwaltungstools: WILDCARDFORMAT

Kurzname des JMS-Verwaltungstools: WCFMT

Programmgesteuerter Zugriff

Setter/Getter

- MQConnectionFactory.setWildCardFormat()
- MQConnectionFactory.getWildCardFormat()

Werte

TOPIC_ONLY
Erkennt nur Platzhalter für Themenebenen, wie sie in der Brokerversion 2 verwendet werden. Dies ist der Standardwert für Verwaltungstools.

Verwenden Sie für Programme "WMQConstants.WMQ_WILDCARD_TOPIC_ONLY".

CHAR_ONLY
Erkennt nur Zeichenplatzhalterzeichen, wie in Broker-Version 1 verwendet.

Verwenden Sie für Programme "WMQConstants.WMQ_WILDCARD_CHAR_ONLY".

Abhängigkeiten zwischen Eigenschaften von Objekten von WebSphere MQ Classes for JMS

Die Gültigkeit einiger Eigenschaften ist von bestimmten Werten anderer Eigenschaften abhängig.

Diese Abhängigkeit kann in den folgenden Gruppen von Eigenschaften auftreten:

- Clienteigenschaften
- Eigenschaften für eine Echtzeitverbindung zu einem Broker
- Exitinitialisierungsbefehle

Clienteigenschaften

Für eine Verbindung zu einem Warteschlangenmanager sind die folgenden Eigenschaften nur dann relevant, wenn TRANSPORT den Wert CLIENT aufweist:

- HOSTNAME
- PORT
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

Wenn TRANSPORT den Wert BIND aufweist, können Sie keine Werte für diese Eigenschaften mithilfe des Verwaltungstools festlegen.

Wenn TRANSPORT den Wert CLIENT aufweist, lautet der Standardwert der Eigenschaft BROKERVER "V1" und der Standardwert der Eigenschaft PORT lautet "1414". Wenn Sie den Wert von BROKERVER oder PORT explizit festlegen, wird Ihre Auswahl von einer späteren Änderung des Werts von TRANSPORT nicht außer Kraft gesetzt.

Eigenschaften für eine Echtzeitverbindung zu einem Broker

Nur die folgenden Eigenschaften sind relevant, wenn TRANSPORT den Wert DIRECT oder DIRECTHTTP aufweist:

- BROKERVER
- CLIENTID
- DESCRIPTION

- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (wird nur für DIRECT unterstützt)
- PORT
- PROXYHOSTNAME (wird nur für DIRECT unterstützt)
- PROXYPORT (wird nur für DIRECT unterstützt)

Wenn TRANSPORT den Wert DIRECT oder den Wert DIRECTHTTP aufweist, lautet der Standardwert der Eigenschaft BROKERVER "V2" und der Standardwert der Eigenschaft PORT lautet "1506". Wenn Sie den Wert von BROKERVER oder PORT explizit festlegen, wird Ihre Auswahl von einer späteren Änderung des Werts von TRANSPORT nicht außer Kraft gesetzt.

Exitinitialisierungsbefehle
Legen Sie keinen Exitinitialisierungsbefehl ohne die Angabe des entsprechenden Exitnamens fest. Die Exitinitialisierungszeichenfolgen lauten wie folgt:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Beispiel: Die Angabe von `RECEXITINIT(myString)` ohne die Angabe von `RECEXIT(some.exit.classname)` hat einen Fehler zur Folge.

Die Eigenschaft ENCODING

Die Eigenschaft ENCODING fasst drei Untereigenschaften in zwölf möglichen Kombinationen zusammen.

Die gültigen Werte, die die Eigenschaft `ENCODING` annehmen kann, werden aus den drei Untereigenschaften konstruiert:

Ganzzahlverschlüsselung
Normal oder umgekehrt

Dezimalverschlüsselung
Normal oder umgekehrt

Gleitkommaverschlüsselung
IEEE (normal), IEEE (umgekehrt) oder z/OS

Die Eigenschaft `ENCODING` wird als Zeichenfolge aus drei Zeichen mit der folgenden Syntax dargestellt:

```
{N|R}{N|R}{N|R|3}
```

Die Zeichen in dieser Zeichenfolge stehen dabei für Folgendes:

- N bedeutet "normal"
- R bedeutet "umgekehrt"
- 3 bedeutet z/OS
- Das erste Zeichen stellt eine *Ganzzahlverschlüsselung* dar
- Das zweite Zeichen stellt eine *Dezimalverschlüsselung* dar
- Das dritte Zeichen stellt eine *Gleitkommaverschlüsselung* dar

Diese Kombinationen stellen zwölf mögliche Werte für die Eigenschaft `ENCODING` bereit.

Es gibt einen zusätzlichen Wert, die Zeichenfolge `NATIVE`, die die entsprechenden Codierungswerte für die Java-Plattform festlegt.

In den folgenden Beispielen sind gültige Kombinationen für ENCODING dargestellt:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

SSL-Eigenschaften von JMS-Objekten

Sie können eine SSL-Verschlüsselung (Secure Sockets Layer) mithilfe der Eigenschaft SSLCIPHERSUITE aktivieren. Mithilfe von verschiedenen anderen Eigenschaften können Sie anschließend die Merkmale der SSL-Verschlüsselung ändern.

Wenn Sie TRANSPORT(CLIENT) angeben, können Sie eine mit SSL (Secure Sockets Layer) verschlüsselte Datenübertragung mithilfe der Eigenschaft SSLCIPHERSUITE aktivieren. Legen Sie als Wert für diese Eigenschaft eine gültige CipherSuite fest, die von Ihrem JSSE-Provider bereitgestellt wird. Dieser Wert muss mit der CipherSpec übereinstimmen, die im Kanal SVRCONN benannt wird, der durch die Eigenschaft CHANNEL angegeben wird.

CipherSpecs (die im Kanal SVRCONN angegeben werden) und CipherSuites (die in ConnectionFactory-Objekten angegeben werden) verwenden jedoch unterschiedliche Benennungsschemas, um dieselben SSL-Verschlüsselungsalgorithmen darzustellen. Wenn ein erkannter CipherSpec-Name in der Eigenschaft SSLCIPHERSUITE angegeben wird, gibt JMSAdmin eine Warnung aus und ordnet die CipherSpec der funktional entsprechenden Cipher-Suite zu. Im Abschnitt SSL-CipherSpecs und -CipherSuites in JMS finden Sie eine Liste der CipherSpecs, die von WebSphere MQ und JMSAdmin erkannt werden.

Wenn eine Verbindung eine CipherSuite verwenden soll, die vom IBM Java-JSSE-FIPS-Provider (IBMJSSEFIPS) unterstützt wird, setzen Sie die Eigenschaft SSLFIPSREQUIRED der Verbindungsfactory auf YES. Der Standardwert dieser Eigenschaft lautet NO, d. h., eine Verbindung kann jede beliebige unterstützte Cipher-Suite verwenden. Wenn SSLCIPHERSUITE nicht festgelegt ist, wird die Eigenschaft ignoriert.

Der Parameter SSLPEERNAME entspricht dem Format des Parameters SSLPEER, der in Kanaldefinitionen festgelegt werden kann. Es handelt sich um eine Liste mit Attributnamen- und Wertepaaren, die durch Kommata oder Semikolons voneinander getrennt sind. Beispiel:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Die Gruppe der Namen und Werte ergibt einen *definierten Namen*. Ausführliche Informationen zu definierten Namen und ihrer Verwendung in WebSphere MQ finden Sie unter Sicherheit.

Das angegebene Beispiel überprüft das identifizierende Zertifikat, das vom Server beim Herstellen der Verbindung vorgelegt wird. Damit die Verbindung erfolgreich hergestellt werden kann, muss das Zertifikat einen allgemeinen Namen haben, der mit 'QMGR.' beginnt. und muss mindestens zwei Organisationseinheitennamen haben, von denen der erste IBM und der zweite WEBSPHERE ist. Bei der Überprüfung wird die Groß-/Kleinschreibung nicht beachtet.

Wenn SSLPEERNAME nicht festgelegt ist, wird diese Überprüfung nicht durchgeführt. SSLPEERNAME wird ignoriert, wenn SSLCIPHERSUITE nicht festgelegt ist.

Die Eigenschaft SSLCRL gibt null oder mehr Server mit Zertifikatswiderrufslisten (CRL; Certificate Revocation List) an. Die Verwendung dieser Eigenschaft erfordert eine JVM mit Java 2 v1.4. Es handelt sich um eine durch Leerzeichen begrenzte Liste mit Einträgen im folgenden Format:

```
ldap://hostname:[port]
```

Optional folgt ein einzelnes "/". Wenn *port* weggelassen wird, wird der LDAP-Standardport 389 angenommen. Wenn die Verbindung hergestellt wird, wird das SSL-Zertifikat, das der Server vorlegt, mit der Liste der angegebenen Server mit Zertifikatswiderrufslisten abgeglichen. Weitere Informationen zur CRL-Sicherheit finden Sie im Abschnitt Sicherheit.

Wenn SSLCRL nicht festgelegt ist, wird diese Überprüfung nicht durchgeführt. SSLCRL wird ignoriert, wenn SSLCIPHERSUITE nicht festgelegt ist.

Die Eigenschaft SSLRESETCOUNT stellt die Gesamtzahl der von einer Verbindung gesendeten und empfangenen Bytes dar, bevor der geheime Schlüssel für die Verschlüsselung erneut vereinbart wird. Dabei ist die Anzahl der gesendeten Bytes die Anzahl vor der Verschlüsselung und die Anzahl der empfangenen Bytes die Anzahl nach der Entschlüsselung. Diese Anzahl an Bytes schließt auch Steuerinformationen ein, die von WebSphere MQ Classes for JMS gesendet und empfangen wurden.

Beispiel: Um ein ConnectionFactory-Objekt zu konfigurieren, das zum Herstellen einer Verbindung über einen für SSL aktivierten MQI-Kanal mit einem geheimen Schlüssel, der nach 4 MB gesendeten und empfangenen Daten erneut vereinbart wird, verwendet werden kann, müssen Sie den folgenden Befehl an JMSAdmin ausgeben:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Wenn der Wert von SSLRESETCOUNT null ist (Standardwert), wird der geheime Schlüssel niemals erneut vereinbart. Wenn SSLCIPHERSUITE nicht festgelegt ist, wird die Eigenschaft SSLRESETCOUNT ignoriert.

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Europe
IBM Europe, Middle East and Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
U.S.A.

Bei Lizenzanforderungen zu Double-Byte-Information (DBCS) wenden Sie sich bitte an die IBM Abteilung für geistiges Eigentum in Ihrem Land oder senden Sie Anfragen schriftlich an folgende Adresse:

Lizenzierung von geistigem Eigentum

IBM Japan, Ltd.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in dieser Veröffentlichung werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Europe, Middle East and Africa
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Um diese so realistisch wie möglich zu gestalten, enthalten sie auch Namen von Personen, Firmen, Marken und Produkten. Sämtliche dieser Namen sind fiktiv. Ähnlichkeiten mit Namen und Adressen tatsächlicher Unternehmen oder Personen sind zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musterprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Musterprogramme kostenlos (d. h. ohne Zahlung an IBM) kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Informationen zu Programmierschnittstellen

Die bereitgestellten Informationen zur Programmierschnittstelle sollen Sie bei der Erstellung von Anwendungssoftware für dieses Programm unterstützen.

Dieses Handbuch enthält Informationen zu geplanten Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zum Abrufen der Services von IBM WebSphere MQzu schreiben.

Diese Informationen können jedoch auch Angaben über Diagnose, Bearbeitung und Optimierung enthalten. Die Informationen zu Diagnose, Bearbeitung und Optimierung sollten Ihnen bei der Fehlerbehebung für die Anwendungssoftware helfen.

Wichtig: Verwenden Sie diese Diagnose-, Änderungs-und Optimierungsinformationen nicht als Programmierschnittstelle, da sie Änderungen unterliegen.

Marken

IBM, das IBM Logo, ibm.com, sind Marken der IBM Corporation in den USA und/oder anderen Ländern. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein.

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Dieses Produkt enthält Software, die von Eclipse Project (http://www.eclipse.org/) entwickelt wurde.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Teilenummer:

(1P) P/N: